Stock Ordering Database

Alex Denby
Computing-Management Studies
2002/2003

Summary

Project Summary

The aim of this project was to develop a web based driven database for a small specialist sports company so that information regarding stock and orders could be stored and searched for by company sales representatives who require remote access to the system. The two main areas of focus were system functionality and usability.

The project minimum requirements were:

- Investigate the appropriate concepts in order to allow an understanding of the user needs for the database.
- Look at possible software tools in order to produce the database and the web based front end for it.
- Examine and evaluate the impact of HCI issues related to the problem so to allow for system usability.
- Develop the database which will satisfy the problem, with the main issues of placing orders and stock control being forefront to the problem.
- Create and apply appropriate procedures to evaluate the solution.

My own personal objectives for this project were:

- Gain experience of using web development tools and improve my own understanding of the programming languages involved.
- Further my knowledge of software development through applying a software development methodology in a practical situation.

Achievements

I have successfully implemented a prototype system that meets the minimum requirements of the project as well as allowing me to achieve my own personal objectives. The system also has other non minimal functionality which I was able to implement within the time constraints of the project schedule.

Table of Contents

1 Introduction 5
1.1 Background to the Problem 5
1.2 Scope of the Project 6
2 Background Research 7
2.1 Systems Development Methodologies 7
2.1.1 Structured Systems Analysis and Design 7
2.1.2 Evolutionary Prototyping 8
2.1.3 Systems Development Life Cycle Model 9
2.1.4 Waterfall Life Cycle Model 10
2.2 Database Management Systems 11
2.2.1 Oracle 11
2.2.2 SQL Server 2000 11
2.2.3 Ingres 11
2.2.4 DB2 12
2.2.5 Access 12
2.3 Web Development Tools 12
2.3.1 Dreamweaver 13
2.3.2 Front Page 2000 13
2.4 Web Page Scripting Language 13
2.5 Human Computer Interaction Issues 14
3 Analysis 15
3.1 Current Systems Procedures 15
3.1.1 Article Details 15
3.1.2 Order Details 15
3.1.3 Customer Details 16
3.2 Need For a New System 16
3.2.1 Data Duplication 16
3.2.2 Data Search Difficulties 17
3.2.3 Update and Delete Problems 17
3.3 User Requirements 17
3.4 Database Requirements 19
4 Database Design 20
4.1 Data Modelling 20
4.1.1 Entity-Relationship Model 20
4.1.2 Entity Types 20
4.1.3 Entity Relationships 21
4.1.4 Attributes 22
4.2 Logical Database Design 23
4.2.1 Mapping the E-R Model into Relational Tables 23
4.2.1.1 Mapping Entity Types 23
4.2.1.2 Mapping 1:1 Relationship Types 23
4.2.1.3 Mapping 1:N Relationship Types 23
4.2.1.4 Mapping M:N Relationship Types 23
4.2.1.5 Mapping Entity Sub-Types 24
4.2.1.6 Mapping Participation of Relationship 24
4.2.2 Relation Schema 24
4.3 Referential Integrity Rules 24

Table of Contents

4.3.1 Entity Integrity 25
4.3.2 Referential Integrity 25
4.4 Database Normalisation 25
4.4.1 Functional Dependencies 25
4.4.2 First Normal Form 26
4.4.3 Second Normal Form 26
4.4.4 Third Normal Form 27
4.4.5 Boyce-Codd Normal Form 28
5 Human Computer Interaction and Systems Interface Design 28
5.1 Human Computer Interaction Guidelines 28
5.2 Readability of Web Pages 28
5.3 Guessibility 29
5.4 Interface Design 29
6 Implementation 31
6.1 Database Implementation 31
6.1.1 Database Relationships 31
6.2 Interface Implementation 32
6.2.1 Web Page Layout 32
6.2.2 User Interface Architecture 32
6.3 SQL and the Database Connection 33
7 Testing and Evaluation 35
7.1 White and Black Box Testing 35
7.2 User Acceptance Testing 35
7.3 Evaluation 36
8 Conclusion 37
8.1 Future Enhancements

Appendix A – Personal Reflection 41
Appendix B – Database Design and Implementation 42
Appendix C – Website Architecture 53
Appendix D – Systems Testing 57
Appendix E – Evaluation 66
Appendix F – User Manual 68

Chapter 1 – Introduction

1.1 Background to the Problem

Blacks Sports specialise in selling specialist sports equipment to high street retailers in England and Wales. They are a company that I worked along side for a period of time while on placement last year. The company doesn't own any retail outlets of its own, nor does it manufacture any of its own products. They are one of a few companies that operate within this niche market segment, with most of their products being developed by sports manufacturers abroad and then imported in to this country. The company has a main Head Quarters which has both the warehouse and the administrative building on the same sight. The company employs sales representatives (reps), each covering a specified area of the company's geographically allocated customer base.

When the company started out, it only provided a service to a relatively small area in England and at this stage in the company life cycle it only utilised a paper based file system.
As the demand for their products grew, it became necessary for the company to employ more staff it order to cope with the increased client base and larger geographical area defined by the new client base. It was also imperative that the company find some larger warehousing premises in order to allow for the increase in stock flow. It was during this transitional time that I became familiar with the company and aware of the problems that they faced, the main one being that it was going to be far too inefficient for the company to continue its administrative tasks in the same way. It was just not going to be viable for the company to continue to employ the same paper based filing system for everything, most importantly, stock allocation and the ordering process.

Unfortunately, the company in question has been unable to fully support this project for business reasons; however I was given initial support in the sense that the company gave me an outline of the problems they faced in trying to design and implement a suitable solution. One of the main problems the company felt it faced was that the reps didn't have accurate, up to date information about stock which they could go to the clients with, so the reps weren't be best equipped to provide the customers with the highest standard of service.
The other major problem that the company was facing as it expanded was to do with the handling of orders. The reps currently take most of the orders for the company and then they send them in to central administration to be processed. This however draws the process out, a flaw highlighted further if orders were to be amended or cancelled. The file based system was just too inadequate,

too slow and generates too many problems such as data duplication and inaccuracies with the orders and consequently stock levels.

It was therefore important that any solution that I generate would eradicate there errors so to allow for a better service to the customers through more accurate and efficient action being permitted to be taken by the reps.

Due to the growth rate of the company it had been decided that the best way forward would be to implement a centrally located stock/ordering database and also some way of remotely accessing it. The remote access being a very crucial element due to the fact that it was going to be used by the field reps the company employed who would be working in different parts of the country. A company intranet had been decided upon as an ideal way forward for a relatively small but expanding company as a way to distribute company information between employees and also as a way of allowing for the necessary remote but secure access to the proposed database. Another reason for the database to be centrally located is that the company were planning to employ and train IT staff to implement and support other business functions that were being developed at the new company site; from here they would also be sufficiently able to maintain the database and the intranet. This all made sense as the system for the reps would be able to run off a centrally located server, which could be easily maintained.

1.2 Scope of the Project

The purpose of this project is to aid the company in overcoming the problems they faced, to produce a solution that will be used by the companies centrally located employees but mainly for the purpose of the sales reps. The solution will be a working database driven website to be accessed via the company intranet. Since this is only a twenty credit project, the solution produced will be a working prototype to the problem which will only be a section of the overall system due to documentation and time constraints. That is the database will only provide functionality in areas relevant to the problem such as available stock and ordering information, it won't look at other areas of the business such as finance or transportation (logistics).

Chapter 2 – Background Research

The purpose of this chapter will be identify and evaluate suitable tools for the development of the system and to consider other aspects relevant to the project design such as Human Computer Interaction (HCI) issues and the appropriate tools to use in the development of the database and the web pages, but firstly I will start by discussing the available software development methodologies that I could follow in the production of this system.

2.1 Systems Development Methodologies

In my experiences at University and while on placement I have been taught the value of approaching all software development in a planned manner. It would therefore be improper to attempt to develop a solution to this problem without adopting a structured systems development methodology; which can be defined as:

"A collection of philosophies, procedures, techniques, tools, and documentation which aid the systems developer with the implementation of a new information system" [3, p.13]

I will firstly outline some of the more well known and adopted methodologies and assess them to see which would best suit my problem so that I can develop a more efficient system.

“Most methods and techniques used by information systems professionals fit one or more of the stages of the life cycle model – with particular emphasis on the analysis and design stages”[2, p119]

For this reason I will be evaluating the Systems Development Life Cycle model along with the waterfall model, Structured Systems Analysis Design Methodology and a more recent methodology called evolutionary prototyping.

2.1.1 Structured Systems Analysis Design Methodology

Similar to the SDLC the Structured Systems Analysis Design Methodology (SSADM) specifies at the beginning what specific tasks need to be addressed. The initial idea behind SSADM was that if the data was modeled correctly then the system would work, later the approach moved away from this ‘hard’ approach, placing more importance on the concept of relationships.

The fact that this methodology has adopted a more soft systems approach has been of great benefit to it. Adopting a softer approach has given way for more human (user) involvement so the system is more likely to be accepted and used as intended.

Some claims for using SSADM are as follows:

- Reduce Life Cycle Development costs through improved analysis and design

- Improve quality of systems delivered
- Improve project management and control
- More effective use of inexperienced staff
- Improve communication between various groups (e.g. User - Analyst)
- Self Documenting

[3]

The positive aspects of this methodology include the emphasis on documentation, and an increase influence on user involvement is systems development. The accompaniment of Computer Aided Systems Engineering (CASE) tools has allowed for a far better quality of documentation as well. The methodology allows for user input to the system and is a lot more robust in accommodating any necessary system design modifications. However, I feel that adopting this systems development methodology would not best serve me in the development of my project as SSADM is a tool used for large scale projects and so could perhaps be too restrictive due to its rigidity and complexities.

2.1.2 Evolutionary Prototyping

A fundamental problem of systems development is the inability of the customer/end user to properly communicate their 'real' requirements to the systems developers. Rapid Evolutionary development by use of prototypes reduces the time scale between the user's expression of need and the developer's demonstration of a (partial) system. This can have several positive effects as outlined below.

Key Benefits of Prototyping:

- More effective user-developer communication because the developers demonstrate what is happening rather than just representing it.
- Due to the rapid development time, the element of uncertainty is lessened, time for problems to occur or circumstances to change are minimised
- Increased ability to deliver desired functionality, achieved through continuous development and refinement via user feedback.
- Supports change, encourages users to modify their requirements as they se fit because it recognises that people operate with incomplete information.

[1]

I believe that to adopt a rapid evolutionary prototyping method could prove extremely useful because of the limited time constraints of the project and the benefits in the final system that can be produced. However, the methodology requires continual user feedback and that the system produced has been continually redefined through this feedback. I know that I am only producing a working prototype, but because I don't have the support of the company users then it would seem futile and imprudent to adopt this method for my software development.

2.1.3 Systems Development Life Cycle

The life-cycle model outlines a set of procedures or stages to follow when developing a system. They are as follows:

1. Preliminary Investigation
2. Systems Analysis
3. System Design
4. System Construction
5. System Implementation
6. Evaluation

[7]

Each step must be visited in order so you can't go back in the sequence to re-visit any step, until you get to evaluation where you can then return to step one.

The System Development Life Cycle (SDLC) methodology has serious limitations as it assumes:

- perfect information
- perfect execution of the system
- That all requirements are identifiable
- That user requirements won't change
- Unchanging Technology

[7]

Changes to requirements are inevitable over time, therefore to use this methodology would be impractical as usually to make changes after implementation would require large changes to the database design [3, p.32]. The methodology only involves users briefly at the beginning of the development process, often resulting in rejection of the system because it fails to meet user requirements. Finally, the documentation produced in the SDLC is computer-oriented and does not allow easy communication between developer and user, again resulting in a system that may

fail to meet user requirements [3, p.30]. So given the limited time available for this development and the other problems highlighted above, to adopt this method just wouldn't be viable.

2.1.4 Waterfall Life Cycle Model

The waterfall model is very closely defined to the SDLC as it adopts a strict procedural approach which must be adhered to as you can't progress to the next step without first completing the current activity. The idea behind the waterfall analogy is to show the ease of which it is go forward, down through the steps and how hard it is to go back up to a previous step. Figure 2.1 illustrates the processes involved. The model is considered to be too simplistic for large projects because developers find the need to repeat steps over and over as the project develops and new ideas are thought up or inconsistencies in the initial design are found. Problems can't just be ignored as this would mean the creation of a system that doesn't work properly, but to accommodate all changes can mean the project will take a lot longer that it should. However, for smaller projects this model may be realisable as the initial design will be simpler and so the risk of problems arising will be reduced. I feel that due to the size of the project and the fact that the user requirements for this prototype development are to remain static that to adopt this method would prove beneficial. A key consideration is for the finished system to be well documented though.

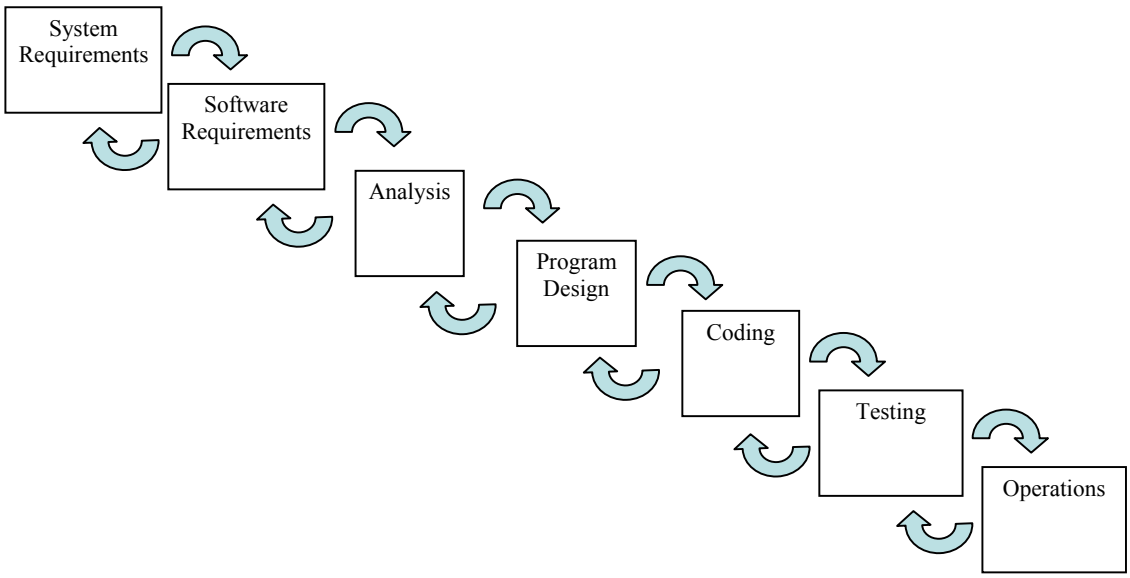


Figure 2.1 The waterfall model of the life cycle [6]

2.2 Database Management Systems

There are several different types of database management system all of which are used to differing degrees within mainstream industry. The most popular products include Oracle, SQL Server 2000, DB2, Ingres and Access 2002. The industry leaders are considered to be SQL Server, Oracle, and DB2 as they hold the majority market share.

There are questions I need to ask and numerous factors that I need to consider when evaluating these various database management systems, one of which is, does the DBMS offer the functionality that is required in order to develop the proposed database? Related to this I must consider the idea that the DBMS has too much functionality and could in fact become a hindrance in developing this relatively small system. Another important factor is cost to the organisation. The company is on a restrictive budget and obviously wants to minimize the costs of utilizing an off-the-shelf package and so careful consideration must be given to this aspect as well.

2.2.1 Oracle

Oracles, Oracle9i package offers a multitude of exceptional features. Due to its ability to power database driven website and the first-rate support it provides it has become the leader in the E-commerce segment of the market. However, I believe that the complexities of supporting the running of Oracle and the cost of purchasing a version for Windows mean that it is beyond the justifiable means of the company.

2.2.2 SQL Server 2000

Microsoft's SQL Server is an extremely popular DBMS, providing a relational database that is highly scalable, and multi dimensional cube technology that addresses the complete spectrum of mainstream database requirements. SQL Server provides extensive database programming capabilities built on web standards [9], and such the data is easily accessible through the web, which is of great relevance and importance to me as the whole concept of this project is to allow the sales reps access to the database through the intranet.

Again, as with Oracle, the main draw backs of SQL Server lie in its cost and its vast functionality which represent too much outlay for the company so I will not be using this DBMS.

2.2.3 Ingres

Computer Associates, Ingres DBMS has experienced large sales increases in recent years but they are still a relatively small player in the market [24]. It is a relatively inexpensive product in comparison to other DBMS but it is still yet to be recognized as an industry standard and I have

never come into contact with it before and as such no nothing of its architecture so to propose to use this as the DBMS for my project development would be rather obtuse.

2.2.4 DB2

IBM's DB2 holds a vast market share, it is an industry standard, it is an industry leader in data management and as such is highly expensive. A factor increased by the various 3rd party products required to create the complete data management package. Because of the complexities of the package the employees who will have to support this would definitely require training which just adds to load of time and money constraints. Again I am not familiar with this product and due to the short term nature of this project I will not be choosing DB2 as my method of implementing the database.

2.2.5 Access

Microsoft Access has been around for 11 years now with the latest version being Access 2002 and it is one of the most popular database development tools. It is extremely simple to use with its Graphical user interface and is quite robust for such a small package. While it doesn't offer the same level of functionality of other DBMS's such as Oracle or SQL Server it does provide ample functionality for this specific project, and I've already discussed the problems, complexities and costs associated with too much functionality. Access is also extremely accessible through a web based front end so this will aid the development process considerably.

Additionally, Blacks already hold a license for Microsoft Office XP professional which includes Access 2002, so there would be no financial costs associated with using this DBMS package. Since I am already familiar with Access and I already have this software available to me I will be using it for the development of my database.

2.3 Wed Development Tools

There are many possible web development packages available and many differ in their functionality and ease of use. Because the end users won't actually have any contact with the web development tool and because the company doesn't currently hold any licenses for such a tool I have quite a bit of freedom in choosing the one I will use, despite this I must still consider the people who will have to maintain the system once it is installed. However, due to the time constraints of the project development I feel that it would best serve me to focus more on the web development tools that I familiar with otherwise I will lose a lot of time acquiring and familiarizing myself with a tool which could hinder the progress of the project. The two tools that

I will evaluate are both popular industry standard packages and they are Macromedia Dreamweaver 3 and Microsoft Front Page 2000.

2.3.1 Dreamweaver

Macromedia Dreamweaver has a well-deserved reputation for being a capable Web design package, and contains support for both graphical design and direct coding of HTML. I have some experience using this software package already, but I did find it quite hard to familiarize myself with, and this could present problems to the staff that would be supporting the system as well as to myself, when it comes to the development.

2.3.2 FrontPage 2000

FrontPage is a powerful program which is very easy to pick up and use, and I've found it an excellent tool when using it before while on placement last year. I consider it to be an ideal tool for creating forms which will be the underlying basis for my web pages. It allows for both direct html coding and for graphical design of pages which is extremely helpful when creating forms. Since the company already operates office XP professional they could easily upgrade to XP developer in order to obtain front page for a much cheaper price. Because I already have a copy of front page and because I feel it is easier to pick up than dreamweaver, a key point for those new to it, I believe that using front page as my web development tool is best option.

2.4 Web Page Scripting Language

In order to develop a database driven website an appropriate development tool need to be selected. There are two main and widely used scripting languages which are PHP and ASP. Before making a choice about which one to use there are various points which need to be considered. The company at the new site is to develop the company intranet through which it is thought this system will eventually run. The intranet will run off a web server which is to run Microsoft's Internet Information Server (IIS) version 5.1 as standard. I have already decided to use Microsoft access for this development so it is imperative that the chosen language tool is compatible with IIS v5 and Access 2002. *"Active Server Pages is an open, compile-free application environment in which you can combine HTML, scripts, and reusable ActiveX server components to create dynamic and powerful Web-based business solutions. Active Server Pages enables server side scripting for IIS with native support for both VBScript and Jscript"* [26]

So ASP supports both vbscript and Jscript where as PHP is its own language which is similar to the Java or C++ languages. On my year out I did a lot of Powerbuilder programming

which is basically VBscript and so I consider myself to be much more able at writing in this language than any other especially since I have only limited C++ skills through the few programming modules I've taken at University and virtually no Java programming experience. ASP is a Microsoft product which and it only runs with Microsoft's IIS where as PHP is portable across many operating systems such as Linux, Microsoft or Solaris [25].
However this is of little consequence because as I've already said Blacks will be running Microsoft's IIS v5.1. Also because ASP is a Microsoft product it is designed to be compatible with other Microsoft products such as IIS and Access so for this reason and the arguments about the actual language I am going to use ASP as my web page scripting language.

2.5 Human Computer Interaction Issues

There is no point in creating a system that meets the user requirements if the users can't use it. In order to address this matter it is imperative that I consider various human computer interaction issues and apply their principles to the interface design. Human-Computer Interaction is the study of the way in which humans operate computers that has the specific goal of ensuring that more usable systems are developed [14, p.5]. The research into HCI has been fully researched so that I could establish which factors are relevant to this prototype development. The findings and their application are fully discussed in Chapter 5 where I demonstrate and justify my web page design.

Chapter 3 – Analysis

In order to create a valid solution the first thing is to fully understand the problem. Without a full comprehension of the current situation and the need for the new system, it will be extremely difficult to identify the user requirements for the new system and subsequently the systems functional requirements.

3.1 Current System Procedures

The next section is a detailed explanation of the current activities undertaken at various stages of the existing system. The majority of the information gathered was through personal contact with company employees while on placement last year but further contact was made via e-mail to the company to in October 2002.

3.1.1 Article Details

The company has a product list of around 130 products which all come in differing size ranges. The company doesn't manufacture any of its own products and so all of the articles have to be bought in. Currently all of these products are stored in an excel spread sheet and all the sales reps have a copy of this product range along with a catalogue of the goods. Each specific product has its own article code so it is uniquely identifiable from all other products. If any new products added to the range or indeed old products removed from the range then the reps are told about this and sent updated product lists. Every month the sales reps are sent stock reports which detail the stock levels of the company so the reps have an idea of what they are able to sell to the customers at any given period. All stock levels are stored on file with a listing of the total quantity of each article and size in the warehouse listed as well as how much of this stock is allocated onto orders.

3.1.2 Order Details

Currently all orders taken by the reps are filled out into a form and sent electronically to the companies central administration to be processed. The order form used by the reps contains details of the customer who placed the order such as the name and delivery address, the product list which basically constitutes a list of required articles, sizes and quantities and the date by which the customer has requested the order. It also has a unique reference on it which is used to identify the order. The administrative staff then use the form to check the stock levels to ensure that there is enough stock available for the order, and pass a copy of the form through to the warehouse so that the order can be prepared for delivery. The stock levels are adjusted to

incorporate the new order and the stock within the warehouse is allocated to the order so that it can't be used for another one. If there isn't sufficient stock then the administration people look to see when the next delivery of the required stock is due to see if it will be possible to make the order by the given date. If not then the new stock will be ordered from their suppliers and the reps will be notified of the problem so that they can go back to their customer with the revised plans. I was assured that this situation does rarely occur because the reps try to get their customers to order in advance so that even if the company doesn't currently have the required articles in stock they can be acquired by the specified delivery date.

All copies of orders are kept on paper because the company currently have no feasible method of storing the records on computer in a suitable way. Any modifications to be made to the orders should be made to the paper based record of the order, with the revised copy being sent to the warehouse and the stock levels adjusted accordingly.

3.1.3 Customer Details

Currently all customer details are stored centrally in excel format and a paper based record of the details is also stored. The sales reps have a complete listing of their own customers stored in a similar fashion and it is the reps responsibility to inform central administration of any new customers and their details and also any changes to current customer information such as the delivery address so that the records can be updated.

3.2 The Need for a New System

As it is clear to see the current system has obvious flaws and this next section will demonstrate the extent of these problems and identify the need for a new system and what the requirements of that new system will be.

3.2.1 Data Duplication

Because information is stored in filing cabinets via the paper based system the information is susceptible to duplication because if a record can't be found then it's easier to create a new record than waste time searching for the desired record. Because of this it's possible to end up with two records for the same order or two records for the same customer but with different information and data anomalies for the same entity are a significant cause for concern.

3.2.2 Data Search Difficulties

A major problem of utilising a paper based system is to do with finding the correct records. Because it isn't viable to keep electronic records of every order the administration must rely on the paper based system but in doing so they leave themselves open to the problems of human error. Order information could be filed incorrectly or because of the problems of data duplication the users may use old data or only part of what is required. Also orders are stored by customer which means that to find a record by anything other than customer such as order due date can be an extremely laborious task.

3.2.3 Update and Delete Problems

Because of the data duplication and search problems updating records can be extremely difficult. This is where data anomalies will develop because if there are duplicate details regarding an order or a customer then it is likely that only one of these records will be deleted or modified successfully so there will be old records within the system that contain incorrect data. The main problem here is that order information isn't updated correctly so the order process is disrupted and customers may get the wrong articles or quantities. Also if order information contains anomalies then this leads to stock level inaccuracies which means that the stock reports sent to the sales reps are inaccurate which creates even more problems for the company.

Overall, the above problems contribute to a whole lack of efficiency within the organisation, the processes are time consuming and the data discrepancies lead to problems with the placing and processing of orders.

From the information I have obtained from the company and the problems highlighted above I have derived a list of user requirements which I feel will best serve the company in eradicating their inefficiencies.

3.3 User Requirements

- A database that can hold information regarding orders, customers and stock
- Allow users to create an order with a unique reference and status
- Allow users to add stock to any order which they create
- Allow users to modify order details
- Allow users to delete order details
- Allow users to view stock details

- It should not be possible to create an order for a customer not yet stored within the system
- It shouldn't be possible to add any stock to an order if that stock isn't stored within the system
- All orders placed on the system must have a valid delivery date
- Allow users to view order details
- Allow users to view customer details
- Allow users to view all orders for each customer
- Allow users to search for orders by delivery date

After going over these issues I concluded that due to the imposed time constraints of the project and the fact that I was only creating a prototype system, it would be essential to evaluate the requirements so that I could implement a feasible solution that would satisfy the minimum requirements of the users. The area of focus is to provide the reps with the ability to have more up to date accurate information. If the reps can see what they are able to order and place their own orders then it will increase their efficiency considerably. With this in mind the following minimal and non minimal requirements for the system were developed:

Minimal Requirements:

1. Storage of order details
    - Allow users to search by unique order reference
    - Allow users to create, cancel and modify orders
2. Storage of company details
    - Allow users to add and modify customer details
    - Allow users to search for company details
3. Search for article details
    - Allow users to search for article details

Non – Minimal Requirements:

1. Advanced order search options
    - Allow users to search for orders by order date or delivery date
    - Allow users to view all orders by company
    - Allow users to search for orders by status

3.4 Database Requirements

From analysing the above user requirements I was able to translate them into minimal database functional requirements. They are as follows:

1. Add an order record
2. Delete an order record
3. Prevent duplications of order records
4. Search for a stock record
5. Modify a stock record on an order
6. Add a stock record to an order
7. Delete a stock record from an order
8. Add a customer record
9. Modify a customer record
10. Search for a customer record
11. Search by article
12. View article information
13. Ensure orders are only created if for existing company
14. Ensure only valid articles are added to orders
15. Search for orders for a customer (non minimal)
16. Provide a status filed to the orders (non minimal)
17. Search for orders by date (non minimal)
18. Search for orders by status (non minimal)

Chapter 4 – Database Design

It is of great importance to this project and indeed any software development involving databases that the database design follows strict processes and guidelines in it development. If the database is not well designed then it will create problems in encapsulating all the requirements gathered through analysis and in keeping data redundancies and anomalies to a minimum.
"…a database must be both well-designed and well-maintained if it is to be of use." [18, p.22]

Additionally, subsequent processes in the development phase are likely to be hindered by a poorly designed database such as the sql used to retrieve information from the database. It is therefore vital that I follow correct data modelling techniques in the database design.

4.1 Data Modelling

A universe of discourse (UoD) can be defined as representing some aspect of the real world. My universe of discourse is concerned with the parts of the company to do with article, customer and order information. Data modelling is the technique used to analyse the UoD and translate it into a graphical model called the Entity-Relationship Model, which in turn can be mapped directly into tables within the database [18, p.33].

4.1.1 Entity Relationship Model

"The ER model describes data as entities, relationships and attributes" [9, p.45]
Entities represent real world objects, whether that object be of physical or conceptual existence. Each entity has attributes which describe the entity and the attributes have values which further describe the entity and formulate a major part of the data stored in the database [9, p.45]. There are several ways of representing the E-R model but for the purposes of this report I will be adopting the Chen notation because it allows attributes to be associated with a relationship, which other notations, such as Rock-Evans [18, p.46], do not allow you to do. I am also most familiar with this notation which will aid me in the design process.

4.1.2 Entity Types

The first stage in data modelling is to identify the entity types. As defined by Mott and Roberts [18, p.33] entities are:

- Something which is in the UoD
- Something which has more than one instance in the UoD

- Something about which we wish to store information

In this case the company has many *orders* about which we want to be able to store similar information. The order entities share the same attributes but are distinguished by their values.
The entity types for my database therefore are:

- Order
- Article
- Customer

In Chen notation the entity types are represented using square boxes. The next phase is to identify the relationships and the degree of the relationships between the entities.

4.1.3 Entity Relationships

Wherever an entity type is referenced by an attribute of another entity type some type of relationship exists [9, p.52]. For example in this case the article attribute of an order refers to the article which is on that order. The relationships can be classified into differing degrees which are to do with the number of participating entity types. The differing degrees are binary, tertiary and so on. However the most common is binary where just two entities participate within the relationship and this is the degree to which all relationships between the entities exist in this database. Further to this is the cardinality ratio associated with each relationship, which denotes the number of relationship instances that an entity can participate in [9, p.57]. The possible cardinality ratios are one-to-one (1:1), one-to-many (1:N) and many-to-many (M:N). For example the relationship between customer and order is one to many as a customer can have many orders but an order can only have one customer. Relationships also have another property which is to do with the entity participation within the relationship. Participation can be *Partial* or *Total* [18, p.39]. Partial participation means that an entity type does not need to participate in a relationship and Total participation means that it is compulsory for an entity type to participate in the relationship [18, p.39]. For example for an order to exist for a company that company must also exist so to hold that order.
In Chen notation relationships are represented by diamond shaped boxes which are connected to the participating entities by diagonal lines. The diagonal lines will be double or single depending on whether the participation within the relationship is total of partial respectively. The cardinality ratio of the relationship is represented by displaying 1, M and N on the diamond shaped boxes.
The relationships between the entities are shown below in table 4.1.

Entity Type 1	Relationship	Entity Type 2	Cardinality Ratio
Customer	Places	Order	1:N
Order	Contains	Article	M:N

Table 4.1

4.1.4 Attributes

The final stage in the ER modelling is to identify all the attributes associated with each entity type. It is the attributes which describe the entity so it's important to correctly identify each attribute. There are several types of attribute which are atomic attributes, composite attributes, single valued attributes, multi valued attributes, stored attributes and derived attributes. The main principle of each is briefly explained here:

Composite attributes can be broken down into more basic attributes where as atomic attributes are already in their simplest form. For example the address field of the company can be broken don into other attributes such as street, city and postcode but city can't be broken down any further. I intend to break all composite attributes into their simplest form as it uncomplicates the database design and will prove important later in the process. Single valued attributes only have one atomic value for an entity but multi valued attributes accept more then one value. All attributes in this case only accept single values for each entity, the importance of which will be demonstrated later. Derived attributes are attributes that can be determined because of their relationship to another attribute. For example, a total orders attribute for each customer could be determined by counting the number of orders that the customer has placed. However, all attributes for the entities are stored attributes.

The attributes for each entity are shown in table 4.2 below:

Entity Type	Attributes
Order	Order_id, order_date, delivery_due_date, status, article_id, article_qty, article_size, customer_id
Customer	Customer_id, customer_name, address_1, address_2, city, postcode
Article	Article_id, article_description, article_size, qty_in_stock

Table 4.2

Once I have all information regarding the entities, their relationships and their attributes I will be in apposition to create my E-R diagram, see Appendix B.

4.2 Logical Database Design

Logical Database Design is the term used to describe the process of mapping the E-R model into the actual relational database schema of the database [9, p.44].

4.2.1 Mapping the E-R Model into Relational Tables

Six rules should be followed when undertaking the mapping of the E-R Model into the actual tables (schema) within the database [18, pp.48-51]. These are:

4.2.1.1 Mapping Entity-types

Each entity-type maps onto a table scheme with associated fields comprising the attributes of each entity-type [18, p.48]. A primary key must be chosen to uniquely identify each record in the table. This prevents record duplication, ensuring that data redundancy does not occur which can cause errors.

4.2.1.2 Mapping 1:1 Relationship-types

Each 1:1 relationship in the E-R Model is represented in the database by posting the primary key of one entity-type into the table scheme of the other entity-type 18, p.48]. There are however no 1:1 entity relationships so this step is not required.

4.2.1.3 Mapping 1:N Relationship-types

1:N relationship-types are dealt with in the same way as 1:1 relationship-types, except that the primary key of the master entity-type is posted into the scheme of the detail entity-type. This effectively means that the schema representing the ‘many’ end of the relationship has the primary key from the ‘one’ end of the relationship posted into it as a foreign key. For example, the primary key of Customer will be posted into the Order schema as a foreign key.

4.2.1.4 Mapping M:N Relationship-types

M:N relationship-types are mapped onto new relationship schemes [18, p.50]. In other words a new table must be created to represent the relationship-type, which will contain the two primary keys of the entity-types participating in the relationship [18, p.50]. So in this case I will need to create a new table that will contain the primary keys from the article schema and the order schema as foreign keys.

4.2.1.5 Mapping Entity Sub-types

Entity sub-types either map onto an attribute-type, whose value specifies the sub-type, or onto a relation scheme separate from the parent type [18, p.50]. Since both of the entity sub-types, Arrival and Departure, have the same attributes and are involved in the same relationship-type, it is viable to map the sub-types using the first method [18, p.50]. Thus an attribute-type 'Arrival or Departure' has been created to indicate whether the entity in question is either of the entity sub-type Arrival or of the entity sub-type Departure. The are no entity sub types within my design so this step can be skipped over.

4.2.1.6 Mapping 'participation' of Relationship-types

The participation property of each relationship-type determines integrity constraints [18, p.51]. If participation is total, every entity of a given entity-type must participate in that relationship [18, p.51]. So for all the entity relationships the primary key posted into each table cannot accept null values. In addition, a value given for the posted primary key (foreign key) in one table must match a value given for the primary key in the other table [18, p.51].

4.2.2 Relation Schema

By applying the above rules I developed the following relation schema.
Order_1 {order_id, customer_id, order_date, delivery_due_date, status}
Order_2 {order_id, article_id, article_size, article_qty}
Article {article_id, article_desc, article_size, qty_in_stock}
Customer {customer_id, customer_name, address_1, address_2, city, postcode}

Notice how there is the extra schema Order_2 which includes both primary keys from Order_1 and Article to overcome the M:N relationship between the two entities.

4.3 Referential Integrity Rules

The relational model created for this database relies on the use of primary and foreign keys. They are required to identify entities to ensure that records are not duplicated and that relationships between entities are correctly defined [18, p.51]. When primary keys are posted into other tables they are called foreign keys in that instance [18, p.51]. Two rules should be adhered to, to ensure that both primary and foreign keys are properly defined. These are Entity Integrity and Referential Integrity.

4.3.1 Entity Integrity

The entity integrity rule states that "no part of a primary key can be null" [9, p.206]. This is to do with the need to ensure that the primary key of an entity has a value otherwise it wouldn't be possible to identify it or relate it to the table that contains the corresponding foreign key.

4.3.2 Referential Integrity

The referential integrity constraint is specified between two relations and is used to maintain consistency among tuples of two relations [9, p206]. In essence this rule ensures that for one table to reference another through its primary key that every instance of the foreign key in the second table must either have an identical value to the value of the primary key or be null. Chapter 6.2 demonstrates how access supports these rules and how I enforced them in order to prevent record duplication which is one of the main functional requirements.

4.4 Database Normalisation

Normalization was first proposed by Codd as a method for putting the relational schema through a series of tests to ensure that it complies with a particular *Normal Form* [9, p.483]. The whole process is aimed at analyzing relations to meet stricter requirements which lead to higher normal forms [9, p.466]. The purpose of normalization is to ensure that two desirable properties are achieved "…(1) minimizing redundancy and (2) minimizing the insertion, deletion and update anomalies…" [9, p.484].

There are four main levels of normal form which are first, second, third and Boyce-Codd normal forms. There are actually a fourth and fifth normal form defined but these deal with multi valued dependencies and join dependencies and it is beyond the scope of this project to discuss them further. Before I discuss the different normal forms though, I must outline the concept of functional dependencies, which form the basis of the theory underlying normal forms. It is worth noting here that the following section is only an outline of the normalization rules and processes, their application of which can be seen in Appendix B, along with the resulting data base design.

4.4.1 Functional Dependencies

"A functional dependency is a constraint between two sets of attributes in the database" [9, p.476]. Basically what it means is that within a schema an attribute may be determined by another attribute or even by a set of attributes. For each of the schema in this database I have determined the following functional dependencies where the left hand side of the dependency forms the key for the schema as it can determine all other attributes in the schema. The right hand side attributes

of the functional dependency are known as non-prime attributes. Further details regarding functional dependencies and other normalization concepts such as relation keys and prime attributes, that are required to be understood for the application the normalization rules, can be found in Appendix B.

Order_1 : order_id → {cust_id, order_date, delivery_due_date, status}
Order_2 : {order_id, article_id, article_size} → article_qty
Article : {article_size, article_id} → {article_desc, qty_in_stock}
: {article_size, article_desc} → {article_id, qty_in_stock}
Customer: cust_id → {customer_name, address_1, address_2, city, postcode }
: postcode → {cust_id, customer_name, address_1, address_2, city}
: customer name → {cust_id, address_1, address_2, city, postcode}

As you can see for two of the tables there is more than one key. This is because there is more than one attribute or set of attributes within the schema that can determine all other non prime attributes. Now these have been defined we can look at the various normal forms

4.4.2 First Normal Form

First normal form (1NF) requires that there be only one value for each attribute of an entity, or putting it another way it only permits "single atomic values" [9, p.485]. There aren't any multi valued attributes present which is why I stated the importance of this earlier. The other issue covered by 1NF is that there can't be any nested relations such as an address attribute consisting of other attributes which formulate the address. This is the other issue I highlighted earlier, to do with the importance of breaking composite attributes down into simple atomic attributes. Because I did this earlier all attributes pass the test for 1NF.

4.4.3 Second Normal Form

"Second normal form (2NF) is based on the principle of full functionally dependency" [9, p488]. The concept of full functional dependency is where a non prime attribute is determined by a key of the relation and removal of any prime attribute from the key results in the relation no longer holding. If the relation was still to hold then the dependency would only be partial and so the schema wouldn't pass 2NF. Where the relation schema key is made up of only one attribute full functional dependency must hold as you can't remove an attribute from the left hand side of the

dependency. All non trivial dependencies are fully functionally dependent so the test for 2NF is passed.

4.4.4 Third Normal Form

Third Normal Form (3NF) is based on the concept of transitive dependency [9, p489]. A functional dependency where X→Y is said to be transitive if there is another set of non prime attributes Z and X→Z and Z→Y. A relation schema R is in 3NF if all non trivial functional dependencies X→A hold in R, and either X is a super key of R or A is a prime attribute or R. [9, p.491.Within my relation schemas there aren't any transitive dependencies so they pass the test for 3NF and they all the non trivial dependencies meet the criteria.

4.4.5 Boyce-Codd Normal Form

Boyce-Codd normal form (BCNF) is almost the same as 3NF but it is slightly stricter. Every relation must be in 3NF and then satisfy one more criteria. The formal definition of BCNF is that "a schema R is in BCNF if whenever a non trivial functional dependency X→A holds in R, then X is a super key of R" [9, p.494] For all my schemas except the Article schema, as demonstrated in appendix ??, for all non trivial dependencies X→A, X is a super key of R so the schema pass the test and so are in BCNF. The Article schema is in 3NF and upon closer analysis I found that doing it in this way would mean to link the orders_2 schema to the article schema would mean that I wouldn't be able to enforce referential integrity on the relationship. By decomposing the schema, a process which is demonstrated in Appendix B, I created the following schemas, which are fully noramalised to BCNF.

Article_1 {article_id, article_size, article_qty}
Article_2 {article_id, article_desc}

It also overcame the problem of referential integrity because the primary key article_id from article_2 can reference the order_2 schema which contains article_id as a foraign key and also the article_1 schema which also contains article_id as a foreign key.

I am now in a position to proceed onto the actual database creation.

Chapter 5 – Human Computer Interaction and Systems Interface Design

The third minimum requirement for this project was to investigate *human-computer interaction* issues (HCI) to ensure that the system is actually usable.
"Usability is central to the human-computer interface, since the whole point of interface design is to produce systems that are easy to learn and which allow users to work efficiently, effectively and comfortably." [12, p.1]

The purpose of this chapter is to investigate and analyse possible HCI issues and see how they will influence the interface design.

5.1 HCI Guidelines

HCI studies have provided some useful guidelines for creating more usable forms. These are:

- *Meaningful title and text label descriptions* – use terms which have explicit meaning.
- *Logical grouping and sequencing of fields* – related fields should be adjacent and aligned
- *Visually appealing layout* – do not overcrowd the screen, ensure alignment is consistent. This allows frequent users to ignore the labels and concentrate on the data-entry fields.
- *Familiar labels and command names* – use terms with which the users are familiar.
- *Consistent Terminology* - if buttons or text boxes perform the same function, label them in the same way
- *Visible boundaries for data-entry fields* – the size of the field box should allow users to anticipate what data should be entered. For example, when entering a date a user could be presented with the following so it is clear as to what they are required to enter and in what order:

  In this case the data the user can enter is already provided so all they need do is select from a list.

[20, pp.263-4]

5.2 Readability of Web Pages

The web pages are to be constructed of html forms as already stated in chapter 2. They are one of the most efficient ways of providing the user with an interface [20, p.262].

Bajaj & Krishnan identify two main factors that influence the readability of a html web page. They are coherence and cognitive overhead [4, p.227]. Coherence can be categorised into two types, local and global. Local coherence is to do with the reader being able to form a mental model of the real word from a hypertext document and global coherence is to do with the degree to which a reader can develop a macro structure across documents [4, p.227]. Application navigation is fundamental to global coherence. If pages jump to others in an illogical manner or they don't allow for backwards navigation then they are going to prevent the user from developing a mental structure of the application [4, p.228]. The key point here is that if the users are able to formulate a mental picture of what is happening on each page and how to maneuver through the application successfully in order to complete specific processes then it will lead to greater system usability.

Cognitive overhead is to do with the limited capacity of human information processing [4, p228] and is mainly determined by user interface adjustment [23, 1995] and low consistency [10, 1995]. The point being that an application that makes use of many varying colours, fonts and layouts will create cognitive overhead to the user, and so they won't be able to give their full attention to the actual system processes, something that is possible when a simpler and more consistent design method is adopted.

5.3 Guessibility

"Guessibility is the term given to the situation where users use previous experience to understand a new system" [8, p.165]. Fields and commands were labelled with terms the users had previously encountered, such as 'delivery date', or 'order_id' for orders so that they would be able to apply previous experience in understanding the workings of the new system. The more a user is able to work out for them self what is going on the faster they will learn about the systems functionality and gain from its benefits.

5.4 Interface Design

With all this in mind I decided that for consistency I would design a template for structuring the layout of all the web pages. I wanted to make use of menu options as they make the page easy to understand and navigate so users can see exactly where each link will take them. I devised a template for the web pages, see figure 5.1. The page would be designed using frames which could be re-used for other pages. The options menu and company information would appear the same on every single web page so only the central part of the page would actually differ between

different web pages. The options menu would consist of the necessary links that the users would need in order to place orders or view specific details.
Because of the influence of color and text on the usability of the web pages I knew that I would have to keep it simple. I decided that the central part of the page would be kept white so the users would give their full attention to the content of it rather than its appearance. I also wanted to distinguish it from the rest of the page so I decided to use colour for the menu and company header. The full interface implementation can be seen in Chapter 6.2.

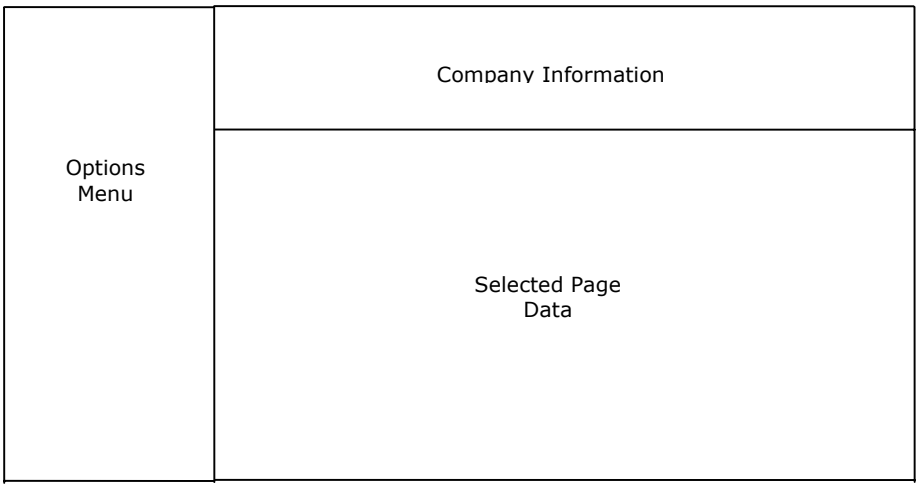


Figure 5.1

Chapter 6 – Implementation

This chapter explains the procedure followed in creating the system prototype. It outlines how I created the database and the user interface drawing on guidelines and information from chapters 4 and 5 respectively, and goes on to explain the method adopted in connecting the two together.

The first thing I had to do was obtain the development tools that I said I would be using for the development. I intended to do all development on my own machine which, while only running the windows 2000 operating system, does have the office XP software installed, which includes Access 2002 which is the same version of Access that Blacks has. Also installed I have Front Page 2000 and some software called Crimson Editor (v3.45) which I intended to use to support my development as it has syntax highlighting options for many programming languages, one of which is ASP. I downloaded version 6.0 of Microsoft's internet explorer which is the latest version of explorer and is distributed with windows XP software. So I now had all the software necessary to develop my project in the way in which I had specified.

6.1 Database Implementation

Firstly I needed to create a database. In Access the graphical user interface (GUI) makes it very easy to create a database. Once that was done, I used the *design view* offered by Access to create the tables from the 5 relation schemas I had created, as shown in chapter 4. I used the Design View so that I could easily specify the field (column) names, the associated data type, if the field was required to take a value or whether it could accept NULL values and also if I wanted to create an index on that field. All indexed fields defined are either primary or foreign keys within their respective table and they are managed accordingly, where by a primary key doesn't accept duplicate values and the foreign is able to. This prevents the duplication of data within the database which was one of the minimal functional requirements. Also all primary and foreign keys must accept values so this ensures that entity and referential integrity can be maintained. The created tables can be seen in Appendix B.

6.1.2 Database Relationships

The next step was to define the relationships between the tables. Access offers an option just for doing this and allows you to specify table and from changing or deleting records in the primary table that would result in unconnected records in a related table [22, p.56]. Figure 6.1 illustrates the tables and the relationships between them. Notice how the relationships between the tables

depict the cardinality ratio, if referential integrity wasn't enforced then Access wouldn't be able to show the relationships in such a way. which tables are linked together, by what attributes and whether referential integrity should be enforced upon the relationship. The referential integrity prevents users from adding records to a table when there is no associated record in the primary.

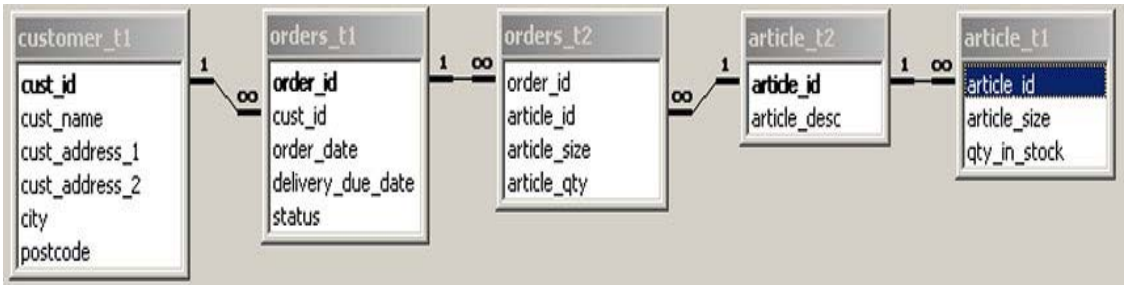


Figure 6.1

The only attribute allowed to accept NULL values is the cust_address_2 attribute of the customer_t1 table. This is because addresses differ and so I couldn't enforce a value to be entered here as it just might not exist in reality. Once the database was created I could proceed to the interface implementation

6.2 Interface Implementation

6.2.1 Web Page Layout

As I proposed in Chapter 5, I intended to use frames for the design of my interface. Frames consist of combining multiple pages together to form a single page. Front page supports frames so I designed all my html documents using front page and created the frame design, shown in figure 6.2, which was used for all the pages. You can clearly see the three pages which constitute the frame. The use of the blue colour clearly differentiates the menu and header from the main part of the frame. I used black text which is clearly legible against both the white and the blue backgrounds. All of the links to the systems pages are down the left hand side in the menu page but the header page also contains a link back to the main page. I kept this link separate as it isn't a link that goes to a functional page as all the other links do.

6.2.2 User Interface Architecture

In Chapter 5, I highlighted the need for the site to be logically designed so that links between pages didn't appear random and confusing to the user. Figure 6.3, shows the higher level links

between the html pages. For a more in depth view of how all the documents are linked together refer to Appendix C.

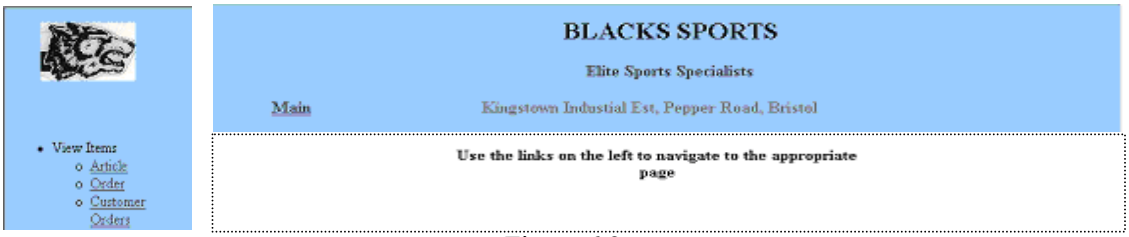


Figure 6.2

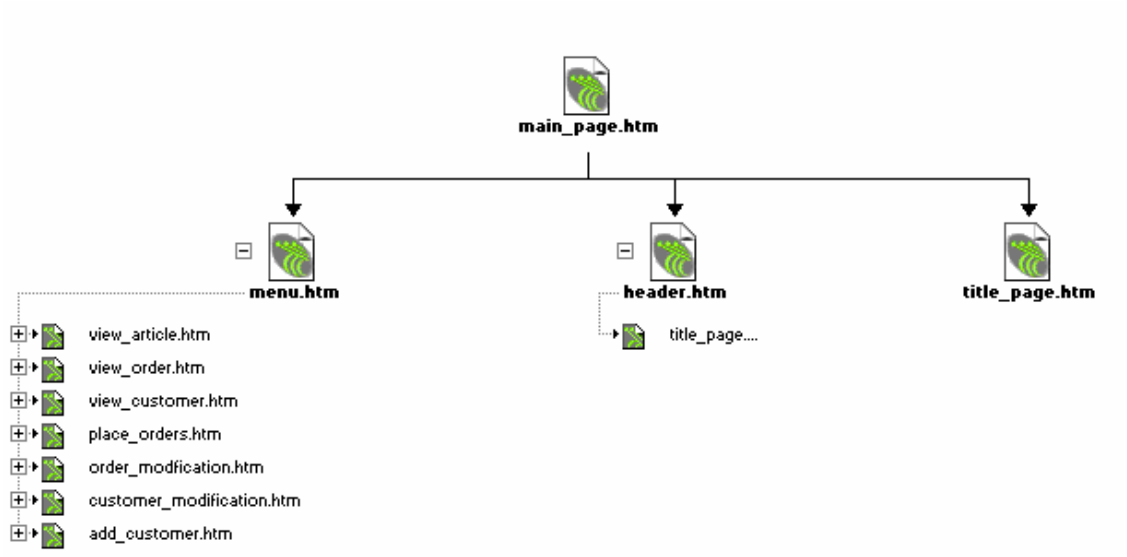


Figure 6.3

6.3 SQL and the Database Connection

Once the pages were designed I could then add the ASP to the html documents. ASP pages use structured query language (SQL) to select modify and delete data from the database. In order to do this the ASP pages must be connected to the database, which is done by creating a connection string, which defines a path to the database. Having done this before in DB32 I knew that instead of defining the connection string in every single ASP file, the best way would be to define the string once in a file and then include a header in each ASP file which points to it. This also allows for better portability because if the database is ever moved then you would only need to change the connection string once. Figure 6.4 shows the connection string used to link the ASP pages to the database.

```
<%
    'Set up the DB connection'
    Dim adoConn, dbPath
    dbPath = "C:\Documents and Settings\Alex Denby\My Documents\Alex Work\Project\project.mdb"
    Set adoConn = Server.CreateObject("ADODB.Connection")
    adoConn.Open "PROVIDER=MICROSOFT.JET.OLEDB.4.0;DATA SOURCE=" & dbPath
%>
```

Figure 6.4

Once a connection to the database has been established it is possible to perform queries on the database. When retrieving any data from the database you must use recordsets which are populated with all the information that you request from the tables in your SQL statement. Within the code must create a recordset, execute a piece of SQL to populate the recordset and then open the recordset to extract the data from it. When updating or deleting data it is possible to just formulate an SQL statement and then execute it. Figures 6.5 demonstrates the coding required to select data from the article table using recordsets and figure 6.6 shows how to delete a record from the orders_t2 table, a process used when modifying an order.

```
    'Creating the Recordset'
    dim sqlSelectArticle
    dim adoRsArticle
    set adoRsArticle = Server.CreateObject("adodb.recordset")
    'defining the sql, executing it and opening the recordset'
    sqlSelectArticle = "Select article_id from article_t1 where article_id = "& article
    adoRsArticle.open sqlSelectArticle, adoConn, 1
```

Figure 6.5

```
    'Specify the SQL code'
    sqlDelete = "Delete * from orders_t2 where order_id = "& order &" and "&_
                "article_id = "& article &" and article_size = '"& size &"'"
    'Execute the code'
    adoConn.Execute(sqlDelete)
```

Figure 6.6

All the SQL I used in my ASP was developed initially developed in Access to make sure it worked as it should and then pasted into the ASP code and adjusted to incorporate the variables within the code.

Overall, through the implementation I was able to develop a prototype system that met with all the users minimal requirements outlined in Chapter 3.3. I was also able to add the additional function of allowing orders to be viewed for each customer which was an additional criterion of the requirements.

Chapter 7 – Testing and Evaluation

The testing process is used to make sure that the system produced meets its initial user and functional requirements. For this project, testing has been broken down in to two specific areas, white and black box testing and user acceptance testing.

7.1 White and Black Box Testing

"White box testing is testing against the implementation and will discover faults of commission, indicating that part of the implementation is faulty" [27]. Due to the nature of white box testing, which requires technical knowledge [11], it was implemented along side the development of the system. Things which were tested for include:

- Data Entry Validation – ensuring only complete and accurate data is entered
- Error Handling – appropriate methods to deal with errors and good user feedback
- That SQL executes as expected – retrieve, delete and update the correct data
- Testing Hypertext documents – ensure all links work as expected

"Black box testing is testing against the specification and will discover faults of omission, indicating that part of the specification has not been fulfilled" [27]. By comparison of the system to the original requirements I will be able to more clearly see where, if any, the faults of omission lie.

Through the continual testing during implementation I was able to piece together a suitable test plan to cover the above issues. It covers some of the more important white box testing issues, because to document all of the testing procedures would be far to complicated, as well as the black box functionality issues of what the system is actually able to do. The test plan can be seen in Appendix D.

7.2 User Acceptance Testing

User acceptance plays a major role in the evaluation of the software. It helps the users to refine their requirements as they can get a better feel for what they want by actually using the system. As I've mentioned before, the lack of support from Blacks in the development of this prototype has meant that they are unable to offer me any help in this area. For this reason I sought the help of two dummy users, who while not understanding the detailed scope of the problem were able to test the systems functionality and provide me with feedback about how they thought it performed.

The users were given a copy of the user manual, see Appendix F, while doing the testing so the could use it to perform specific tasks. Because I used dummy users a few key points need to be taken into consideration when evaluating the findings:

- The users have little understanding of the problem and as such the system requirements.
- The lack of knowledge leads to no pre-conceptions about how they feel the system should function.
- Lack of current systems procedure means that the terminology deliberately used within the interface will be of little meaning so the impact of systems guessibility is somewhat reduced.

The user acceptance testing process is detailed in Appendix D.

7.3 Evaluation

I based much of my evaluation on the guidelines established by McCall et al [17]. His guidelines for evaluating software developments focus on the functional correctness, usability and adaptability of the software product, which are all extremely important to the quality of the software product.

The criteria which I felt to bear most relevance to this project are:

- **Correctness:** The extent to which the development satisfies the user requirements
- **Reliability:** The extent to which the system performs its functions without failure
- **Efficiency:** The extent to which the database improves on the current paper-based system
- **Usability:** The extent to which the system is usable
- **Maintainability:** The effort required to fix errors or update functions in the system

I missed out criteria such as integrity, which covers issues concerning authorised access to the system, because it is assumed that the system would be accessed through the company intranet and to investigate this would be well beyond the scope of the project.

I cross referenced the criteria against the results from the system testing and the user acceptance testing so to provide a broad scope for the evaluation. Within the evaluation I also included some of my own thoughts and personal reflections about the project from the developer's perspective. Appendix E details the evaluation findings.

Chapter 8 – Conclusion

The evaluation results show that overall the development of the prototype system for a stock ordering system to be fairly successful. The minimum user requirements developed through systems analysis were exceed, the software developed was done through following strict design processes after lengthy research had been conducted and the prototype produced was evaluated using specified guidelines. However, in concluding this report I will assess the project outcomes against the minimal requirements outlined at the very beginning.

As outlined at the beginning of the report, the project minimum requirements were:

- Investigate the appropriate concepts in order to allow an understanding of the user needs for the database.
- Look at possible software tools in order to produce the database and the web based front end for it.
- Examine and evaluate the impact of HCI issues related to the problem so to allow for system usability.
- Develop the database which will satisfy the problem, with the main issues of placing orders and stock control being forefront to the problem.
- Create and apply appropriate procedures to evaluate the solution.

The current systems analysis included Chapter 3 was dedicated to outlining the current systems procedures and to establish the needs for the new system which allowed me to identify the user requirements of the system. Therefore it meets the first minimal requirement.

Chapters 2, 4 and 5 were concerned with extensive background research in to suitable DBMS, web page developments tools, scripting languages, database design and HCI issues relating to the database design, the user interface design and their development. Many practical issues were taken into consideration when evaluating the extensive research and designing the system and I believe that the project meets with the second and third criteria.

The system prototype was then developed, using the carefully designed structures, which surpassed the minimal user requirements and so criteria four of the project requirements has been met.

Finally I adopted various testing techniques to assess the system functionality and usability which were evaluated against a set of software development evaluation guidelines which had been refined to have more relevance to the project. So the last minimal requirement has also been

achieved. Overall the project has proved successful but there are many areas for future enhancement and development.

8.1 Future Enhancements

As the system is only a prototype there still large scale possibilities for how the system could develop. The work done here has only scratched the surface of the entire systems flow, future enhancements could be to encompass other parts of the system, such as logistics or the ability for users to order stock from suppliers, not just place orders for customers. A fully integrated system would reap huge benefits the company and maybe this is where the systems ultimate goal could lie.

Whichever direction the system now moves in, as this project has shown to me, the development will need to be properly researched, justifiably designed, implemented structurally and fully tested and evaluated so achieve high quality and be of functional usage.

Bibliography

[1] Arthur L.J, 1992, Rapid Evolutionary Development: Requirements, Protyping & Software Creation

[2] Avgerou and Cornford, 1993, Developing Information Systems: Concepts, Issues and Practice

[3] Avison and Fitzgerald, 1995, Information Systems Development : Methodologies, Techniques, And Tools

[4] Bajaj A & Krishnan R, 1998, Analyzing Models for current World Wide Web Applications Using a Classification Space and Usability Metrics

[5] Barrier T, 2002, Human Computer Interaction Development and Development

[6] Boehm, 1998

[7] Bull M, 1989, Systems Development using structured techniques

[8] Dix A, Finlay J, Abowd G & Beale R, 1998, Human-Computer Interaction

[9] Elmasri R. and Navathe S.B., 2000, Fundamentals Of Database Systems (3rd Edition), Addison Wesley Longman

[10] Garzotto F, Mainetti L & Paolini P, 1995, Hypermedia Design, Analysis and Evaluation Issues, Communications of the ACM, 38(8), 74-86

[11] Heathcote, P M(1996) Program Production and Testing in: Computing- An active learning approach, 3 rd Edition, Letts pp. 80 – 85

[12] Hill S, 1995, A Practical Guide to the Human-Computer Interface

[13] Kendall & Kendall, 1995, Systems Analysis and Desigh 3rd Ed

[14] Kirakowski J, Human Computer Interaction: from voltage to knowledge, Chartwell-Bratt, 1988

[15] Krutchen Philippe, The Rational Unified Process: An Introduction (2nd Edition), Addison-Wesley, 2000

[16] Lantz K.E, 1986, The Prototyping Methodology

[17] McCall J., Richards P., and Walters G., Factors in Software Quality, Vol. I, AD-A049014/015/055, November 1997

[18] Mott P, and Roberts S, *DB11: Introduction to Database*s, Module Notes and Coursework, School of Computing, University of Leeds, 1999-2000

[19] Roberts S, *DB21: Database Principles and Practic*e, Module Notes, School of Computing, University of Leeds, 2001-2002

[20] Shneiderman B, 1998, Designing the User Interface: Strategies for Effective Human-Computer Interaction, 3rd Ed.

Bibliography

[21] Sommerville, 2001, Software Engineering 6th Ed

[22] Stephen Moria, 2000, Teach Yourself Access 2000

[23] Thuring M, Hannemann J, Hake J.M, 1995, Hypermedia and Cognition: Designing for Comprehension. Communications of the ACM, 38(8), 57-66

[24] http://www3.ca.com/Solutions/Product.asp?ID=1013, November 2002

[25] http://php.weblogs.com/php_asp_7_reasons, 15th February 2003

[26] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnasp/html/aspover.asp, 24th April 2003

[27] http://www.scism.sbu.ac.uk/law/Section5/chap3/s5c3p23.html, 24th April 2003

Appendix A – Personal Reflection

At the start of the project my own personal objectives were:

- Gain experience of using web development tools and improve my own understanding of the programming languages involved.
- Further my knowledge of software development through applying a software development methodology in a practical situation.

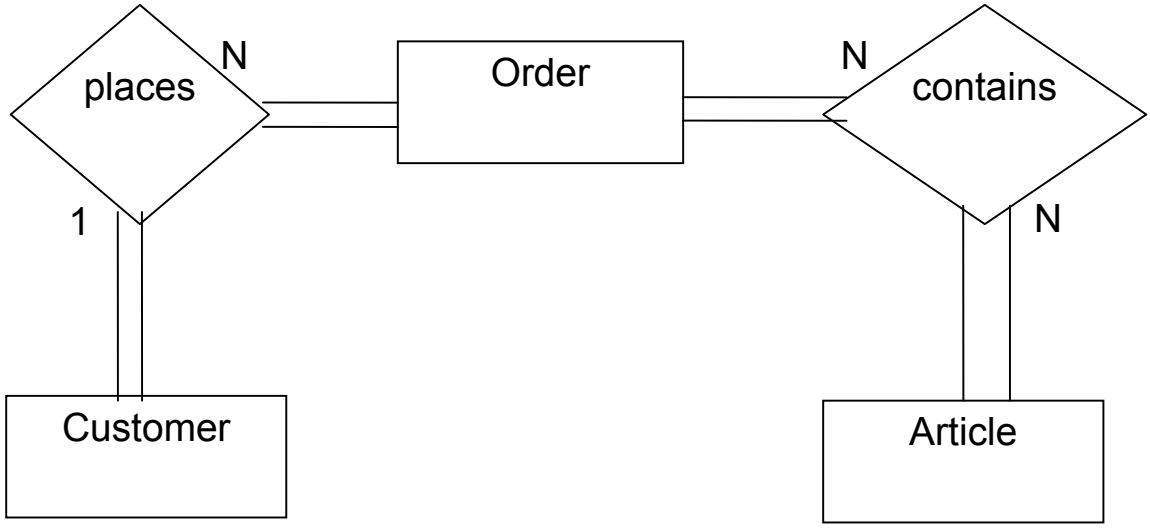
I believe that I have achieved them and more. The experience of programming in ASP was one that I found trick at first but most rewarding, and by the end of the project I was able to easily identify scripting errors and correct problems so I feel I have learned a lot, much of it coming through the internet and the understanding of third party code.
Probably even more valuable was the knowledge I have gained through the research into the project and the application of guidelines and structured methods which I discovered and followed in the creation of the software. I had developed software in real life situations before while on my year out and as such had followed structured methods, but these methods were more closely based around the evolutionary system where user feedback was often gained so to redefine the systems. Due to the lack of user feedback in this project I had to investigate new ways of evaluating the software, and I believe I have really learned from it.

My worst down fall on this project was sticking to the schedule. In the beginning I had under estimated the task that I was undertaking, something which has really been forced home during the last few weeks. My project management skills have certainly been improved and I have definitely learned through this experience as the pressures I put on my self were unnecessary and not something which I would like to go through again.

Appendix B – DataBase Design and Implementation

This appendix is here to supplement the information regarding database design and implementation, which is described in chapters 4 and 6 respectively.

Chapter 4 describes the process of how to determine entities, attributes and relationships from the Universe of Discourse in order to produce an E-R diagram. The E-R diagram derived from the entities and relationships in Section 4 can be seen below. The attributes for the entities are not shown in the diagram.



Section 4 goes on to describe the processes involved in mapping from the E-R model to the actual database relation schema which then leads into a brief account of the normalisation process applied to these schema. The Next part of this appendix describes the normalisation process in full and evaluates each of the schemas I developed against it.

DataBase Normalisatoin

Normalization was first proposed by as a method for putting the relational schema through a series of tests to ensure that it complies with a particular *Normal Form* [9, p.483]. The whole process is aimed at analysing relations to meet stricter requirements which lead to higher normal forms [9, p.466]

The purpose of normalisation is to ensure that two desirable properties are achieved "…(1) minimizing redundancy and (2) minimizing the insertion, deletion and update anomalies…" [9,

484]. There are four main levels of normal form which are first, second, third and Boyce-Codd normal forms. There are actually a fourth and fifth normal form defined but these deal with multi valued dependencies and join dependencies and it is beyond the scope of this project to discuss them further. Before I discuss the different normal forms though, I must outline the concept of functional dependencies, which form the basis of the theory underlying normal forms.

Functional Dependencies

"A functional dependency is a constraint between two sets of attributes in the database" [9, .476]. Basically what it means is that within a schema an attribute may be determined by another attribute or even by a set of attributes. In understanding normal forms there are a few key concepts that need to be defined.

Trivial and Non Trivial Dependencies

Where an attribute is used to determine itself the dependency is said to be trival. E.g order_id→order_id. All other functional dependencies are said to be non trivial and it is this type of functional dependency that has an impact on the normalisation level of a relation schema.

Keys and Superkeys

A key is defined as a chosen subset of the attributes of a relation scheme that has the following properties. Firstly that no two tuples in any relation instance have the same combination of values from those attributes and secondly that the subset of the attributes is minimal in this respect [9]. That second point means that if two rather than three attributes can determine all the other non prime attributes then the key should consist of those two attributes and the third attribute is actually non prime as well.

A super key is defined as any set of attributes that contain a key. So essentially a super key satisfies the first property of being a key but it isn't necessarily minimal, so a key can be a super key of the relation, but a super key cannot be a key. [19, p7-5]

e.g order_id → {cust_id, order_date, delivery_due_date, status} So {order_id} is the key but if we said: {order_id , cust_id} → {order_date, delivery_due_date, status}

then {order_id, cust_id} would be a superkey of the relation scheme.

Appendix B

Prime and Non Prime Attributes

For an attribute to be prime it must be a member of a key of the relation schema, all other attributes are called non-prime attributes.

For each of the schema in this database I have determined the following functional dependencies where the left hand side of the dependency forms the key for the schema as it can determine all other attributes in the schema.

The functional dependencies of the relation schema:

Order_1 : order_id → {cust_id, order_date, delivery_due_date, status}

Order_2 : {order_id, article_id, article_size} → article_qty

Article : {article_size, article_id} → {article_desc, qty_in_stock}

: {article_size, article_desc} → {article_id, qty_in_stock}

Customer: cust_id → {customer_name, address_1, address_2, city, postcode }

: postcode → {cust_id, customer_name, address_1, address_2, city}

: customer_name → {cust_id, address_1, address_2, city, postcode}

For two of the schema there is more than one key. This is because there is more than one attribute or set of attributes within the schema that can determine all other non prime attributes.

First Normal Form

First normal form (1NF) requires that there be only one value for each attribute of an entity, or putting it another way it only permits "single atomic values" [9, p.485]. The other issue covered by 1NF is that there can't be any nested relations such as an address attribute consisting of other attributes which formulate the address.

Second Normal Form

"Second normal form (2NF) is based on the principle of full functionally dependency" [9, p488]. The concept of full functional dependency is where a non prime attribute is determined by a key of the relation and removal of any prime attribute from the key results in the relation no longer holding. If the relation was still to hold then the dependency would only be partial and so the schema wouldn't pass 2NF. Where the relation schema key is made up of only one attribute full functional dependency must hold as you can't remove an attribute from the left hand side of the dependency.

Appendix B

Third Normal Form

Third Normal Form (3NF) is based on the concept of transitive dependency [9, p489]. A functional dependency where X→Y is said to be transitive if there is another set of non prime attributes Z and X→Z and Z→Y. A relation schema R is in 3NF if all non trivial functional dependencies X→A hold in R, and either X is a super key of R or A is a prime attribute or R. [9, p.491

Boyce-Codd Normal Form

Boyce-Codd normal form (BCNF) is almost the same as 3NF but it is slightly stricter. Every relation must be in 3NF and then satisfy one more criteria. The formal definition of BCNF is that "a schema R is in BCNF if whenever a non trivial functional dependency X→A holds in R, then X is a super key of R"[9, p.494]

Applying the Rules to each Schema

Order_1

Schema: Order_1 {order_id, customer_id, order_date, delivery_due_date, status}
Trivial Dependencies: order_id → {customer_id, order_date, delivery_due_date, status}
Key attribute: {order_id}

1NF

There are no nested relations and each attribute only needs to take a single value so this passes the test for 1NF

2NF

For all non trivial dependencies there is only one attribute on the left hand side which is the key and so the non prime attributes must be fully functionally dependent on the key as you can't remove an attribute from the left hand side of the dependency so the test for 2NF is passed.

3NF

Within this schema there aren't any non-prime attributes functionally dependent on other non prime attributes so there can't be any non-prime attributes transitively dependent upon the key so the test for 3NF is passed.

Appendix B

BCNF

In the non-trivial functional dependency outlined above the left hand side of the dependency is a super key of the schema so it does pass the test for BCNF.

Order_2

Schema: Order_2 {order_id, article_id, article_size, article_qty}

Non Trivial Dependencies: {order_id, article_id, article_size} → article_qty}

Key: {order_id, article_id, article_size}

1NF

There are no nested relations and each attribute only needs to take a single value so this passes the test for 1NF

2NF

There is only one non prime attribute and if you were to remove an attribute from the key in the non trivial functional dependency then it would no longer hold.

i.e {order_id, article_id} → article_qty doesn't hold

{order_id, article_size} → article_qty doesn't hold

{article_id, article_size} → article_qty doesn't hold

So in this relation schema passes the criteria for 2NF

3NF

There is only one non prime attribute so it can't be transitively dependent on the key of the schema as there are no other non-prime attributes to be functionally dependent on. Also within the non trivial functional dependency X→A, X is a super key of R. That is that because the LHS of the non trivial dependency is the key then it can be considered as the super key so the schema meets the criteria for 3NF as well.

BCNF

Because the schema is in 3NF and the LHS of the non trivial dependency is a superkey, as demonstrated above then it is clear to see that this relation schema is in BCNF.

Appendix B

Customer

Schema: Customer {customer_id, customer_name, address_1, address_2, city, postcode}

Non trivial Dependencies: cust_id → {customer_name, address_1, address_2, city, postcode }

: postcode → {cust_id, customer_name, address_1, address_2, city}

: customer_name → {cust_id, address_1, address_2, city, postcode}

Key: {cust_id}, {postcode}, {customer_name}

1NF

There are no nested relations and each attribute only needs to take a single value so this passes the test for 1NF

2NF

For all non trivial dependencies there is only one attribute on the left hand side which is the key and so the non prime attributes must be fully functionally dependent on the key as you can't remove an attribute from the left hand side of the dependency so the test for 2NF is passed.

3NF

The isn't any transitive dependency between the key and any non prime attributes and for all non trivial dependencies the LHS of the dependency is a super key of the schema so the schema is in 3NF.

BCNF

Because the schema is in 3NF and for all non trivial dependencies X→A, X is a super key of the schema the schema is in fact in BCNF.

Appendix B

Article

Schema: Article {article_id, article_desc, article_size, qty_in_stock}

Non Trivial Dependencies: {article_size, article_id} → {article_desc, qty_in_stock}
{article_size, article_desc} → {article_id, qty_in_stock}
{article_id} → {article_desc}
{article_desc} → {article_id}

Key: {article_size, article_id}, {article_size, article_desc}

1NF

There are no nested relations and each attribute only needs to take a single value so this passes the test for 1NF

2NF

Because article_id and article_desc can functionally determine each other then it may appear as though there is a problem with the full functionality of the first two non trivial dependencies as you could remove article size from the LHS of the dependency and article_desc or article_id can still be determined. However the rule of full functionality applies to non prime attributes being fully functionally dependent on the relation key and so because article_id and article_id form part of a key then full functionality is in fact not breached. Therefore the relation schema is in 2NF

3NF

The relation scheme is indeed in 3NF because in the top two functional dependencies the LHS is a super key of the relation and in the bottom two functional dependencies the RHS is a prime attribute as it forms part of a key.

BCNF

From the showing above to prove 3NF we can see that the relation schema does not meet the criteria for BCNF. This is because to be in BCNF the non trivial functional dependencies X→A, X must be a super key of the relation and neither article_id or article_desc are a superkey on their own so the schema is in 3NF but not BCNF.

Because of this I decided to investigate the impact of having this schema in 3NF. I looked at the relationship between article and orders_2 and found that if I was to convert these schemas into

tables now, I wouldn't be able to enforce referential integrity between the two tables. For this reason I decided to decompose the schema into two separate schemas so to be in BCNF.
Article {article_id, article_desc, article_size, qty_in_stock} became:

Article_1 {article_id, article_size, qty_in_stock}
Article_2 {article_id, article_desc }

By decomposing the schema into two relational schemas I overcame the problem of referential integrity because the primary key article_id from article_2 can reference the order_2 schema which contains article_id as a foreign key and also the article_1 schema which also contains article_id as a foreign key. So then I ran both of these through the normalisation process to ensure they were normalised properly.

Article_1
Schema: Article_1 {article_id, article_size, qty_in_stock}
Non trivial Dependencies: {article_id, article_size} → qty_in_stock
Key: {article_id, article_size}

1NF
There are no nested relations and each attribute only needs to take a single value so this passes the test for 1NF

2NF
If I was to remove an attribute from the non trivial functional dependency then the dependency would no longer hold as neither article_id or article_size on their own can determine qty_in_stock. Therefore the relation schema is in 2NF

3NF
The relation scheme is indeed in 3NF because in the top two functional dependencies the LHS is a super key of the relation.

BCNF
Because the schema is in 3NF and for the reasons its in 3NF the schema also satisfies the criteria for BCNF.

Appendix B

Article_2

Schema: Article_2 {article_id, article_desc}

Non trivial Dependencies: article_id → article_desc

: article_desc → article_id

Key: article_id

: article_desc

1NF

There are no nested relations and each attribute only needs to take a single value so this passes the test for 1NF

2NF

In both the non trivial dependencies there is only one LHS attribute which is a key of the relation schema so you can't remove an attribute from the LHS. Also seen as there are no non prime attributes there can't be any partial dependencies so the schema is in 2NF.

3NF

The relation scheme is indeed in 3NF because in the top two functional dependencies the LHS is a super key of the relation.

BCNF

Because the schema is in 3NF and for the reasons its in 3NF the schema also satisfies the criteria for BCNF.

So I now have 5 schemas which are all in BCNF and allow me to impose referential integrity upon their relationships. The schemas are:

Order_1 {order_id, customer_id, order_date, delivery_due_date, status}

Order_2 {order_id, article_id, article_size, article_qty}

Customer {customer_id, customer_name, address_1, address_2, city, postcode}

Article_1 {article_id, article_size, qty_in_stock}

Article_2 {article_id, article_desc}

Appendix B

Because I've ensured a suitable level of normalization for my relation schema I am now ready to turn the schema in to tables and create the database. The next section shows my tables and the final database design.

Database Implementation

The following tables were created in Access using the Design View so that I could easily specify the column (field) names, the associated data type, if the field was required to take a value or whether it could accept a NULL value and also if I wanted to create an index on that column. All indexed fields are either primary or foreign keys within the table and they are indexed accordingly, where by a primary key doesn't accept duplicate values and the foreign is able to. This prevents the duplication of data within the database which was one of the minimal functional requirements. Also all primary and foreign keys must accept values so this ensures referential integrity is maintained.

Orders_t1

Field Name	Data Type	Value Required	Indexed
Order_id	Number	Yes	Yes(No Duplicates)
Cust_id	Number	Yes	Yes(Duplicates OK)
Order_date	Datetime	Yes	No
Delivery_due_date	Datetime	Yes	No
Status	Text	Yes	No

Orders_t2

Field Name	Data Type	Value Required	Indexed
Order_id	Number	Yes	Yes(Duplicates OK)
Article_id	Number	Yes	Yes(Duplicates OK)
Article_size	Text	Yes	No
Article_qty	Number	Yes	No

Appendix B

Customer_t1

Field Name	Data Type	Value Required	Indexed
Cust_id	Number	Yes	Yes(No Duplicates)
Cust_name	Text	Yes	No
Cust_address_1	Text	Yes	No
Cust_address_2	Text	No	No
City	Text	Yes	No
Postcode	Text	Yes	No

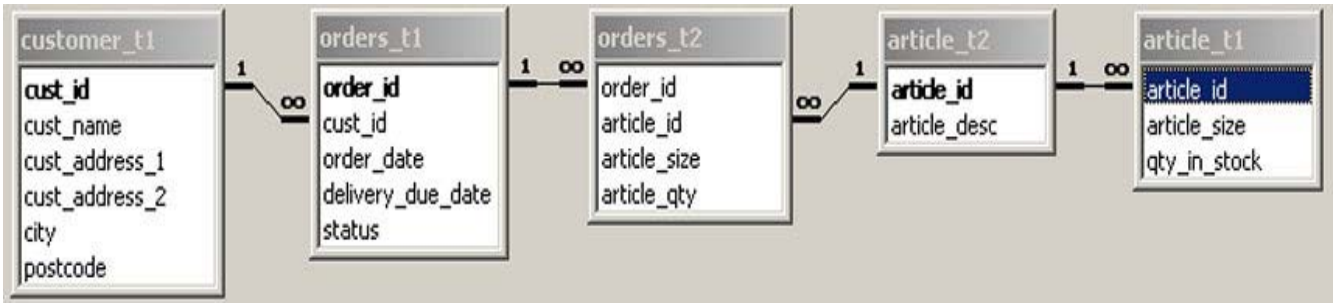
Article_t1

Field Name	Data Type	Value Required	Indexed
Article_id	Number	Yes	Yes(Duplicates OK)
Article_size	Text	Yes	No
Qty_in_stock	Number	Yes	No

Article_t2

Field Name	Data Type	Value Required	Indexed
Article_id	Number	Yes	Yes(No Duplicates)
Article_desc	Text	Yes	No

The following diagram illustrates the relationships between the tables.

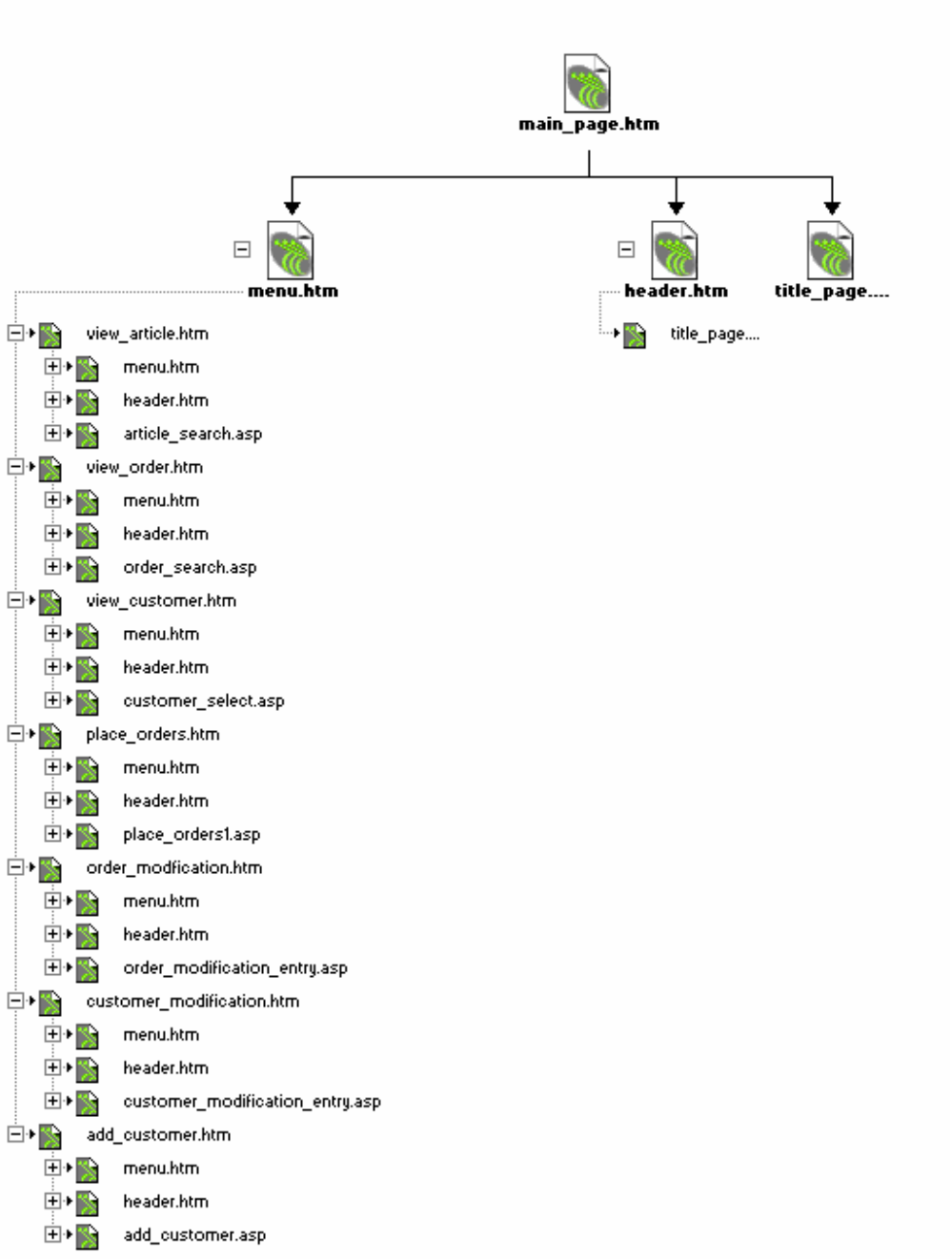


Notice how the relationships between the tables depict the cardinality ratio, if referential integrity wasn't enforced then Access wouldn't be able to show the relationships in such a way.

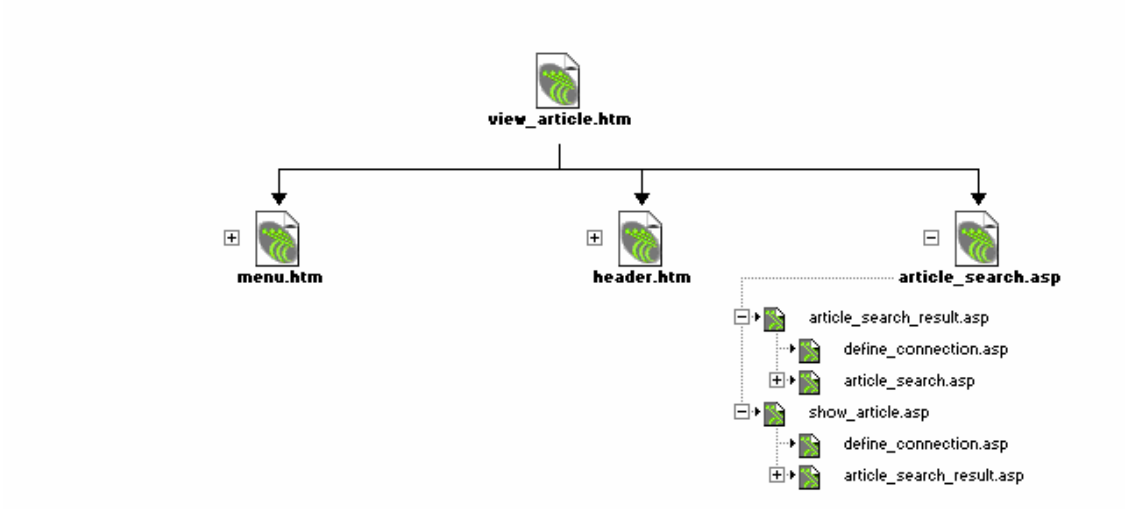
At this point after following all the necessary development processes, the database can be considered as completed.

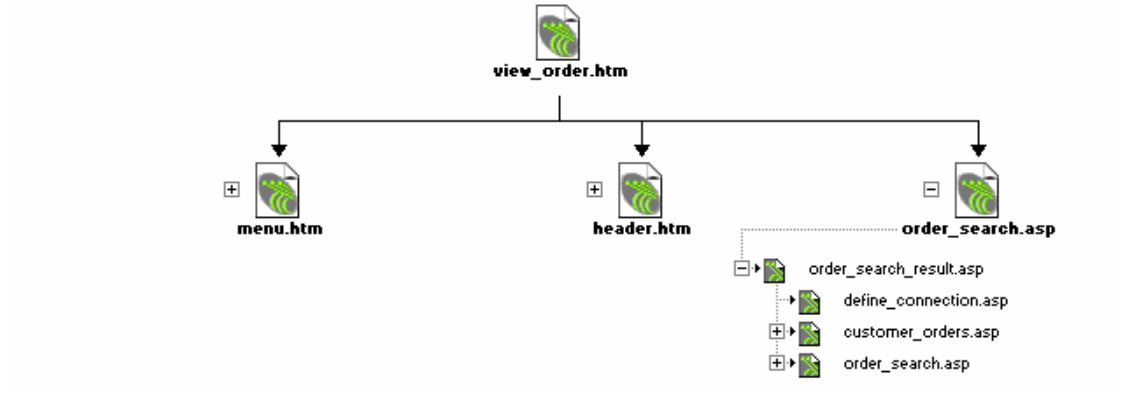
Appendix C – Web site Architecture

The purpose of this appendix is to illustrate the web site architecture briefly discussed in section 6. The following diagrams display the links between all the pages and demonstrate the logical design of the system.

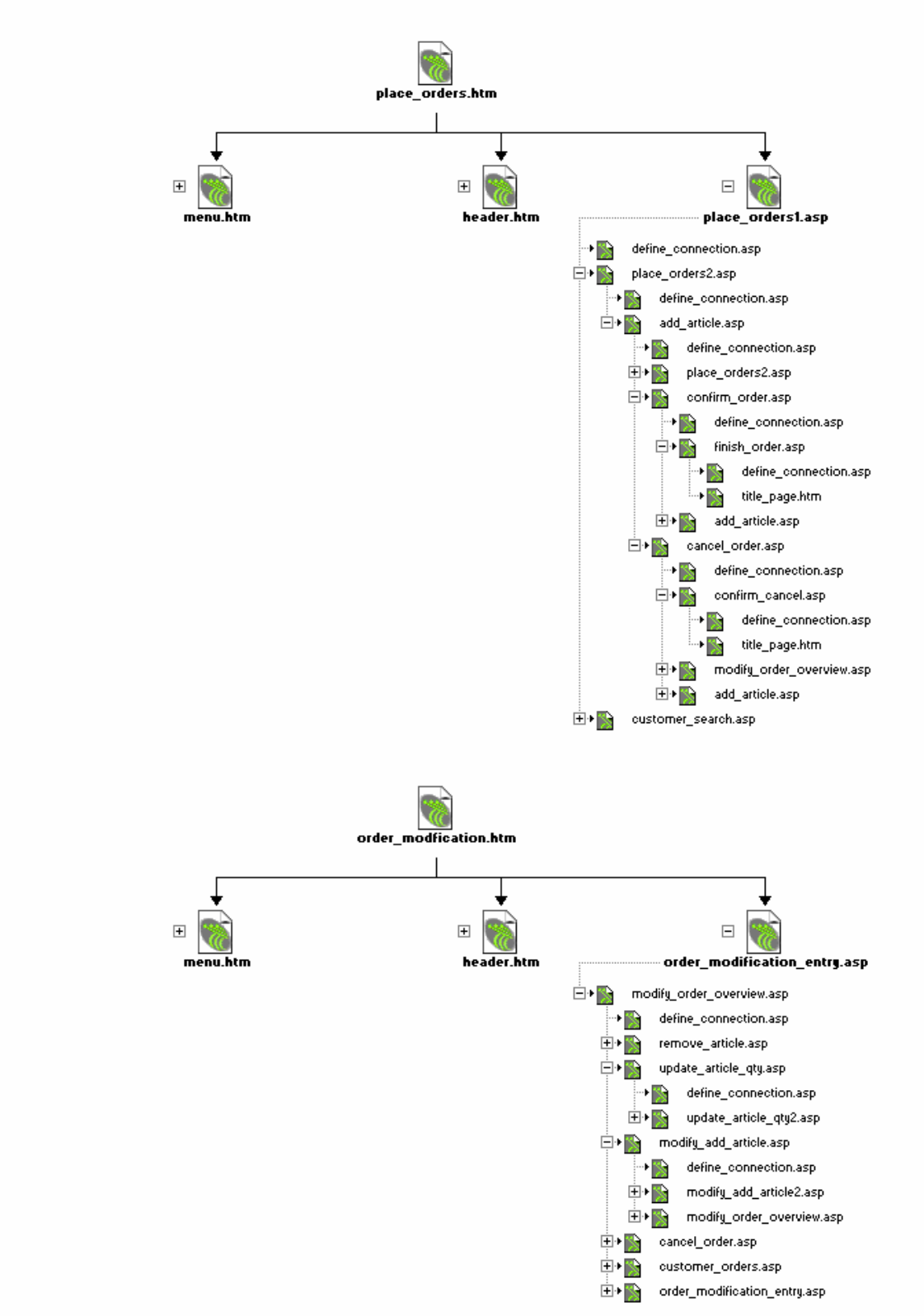


Appendix C

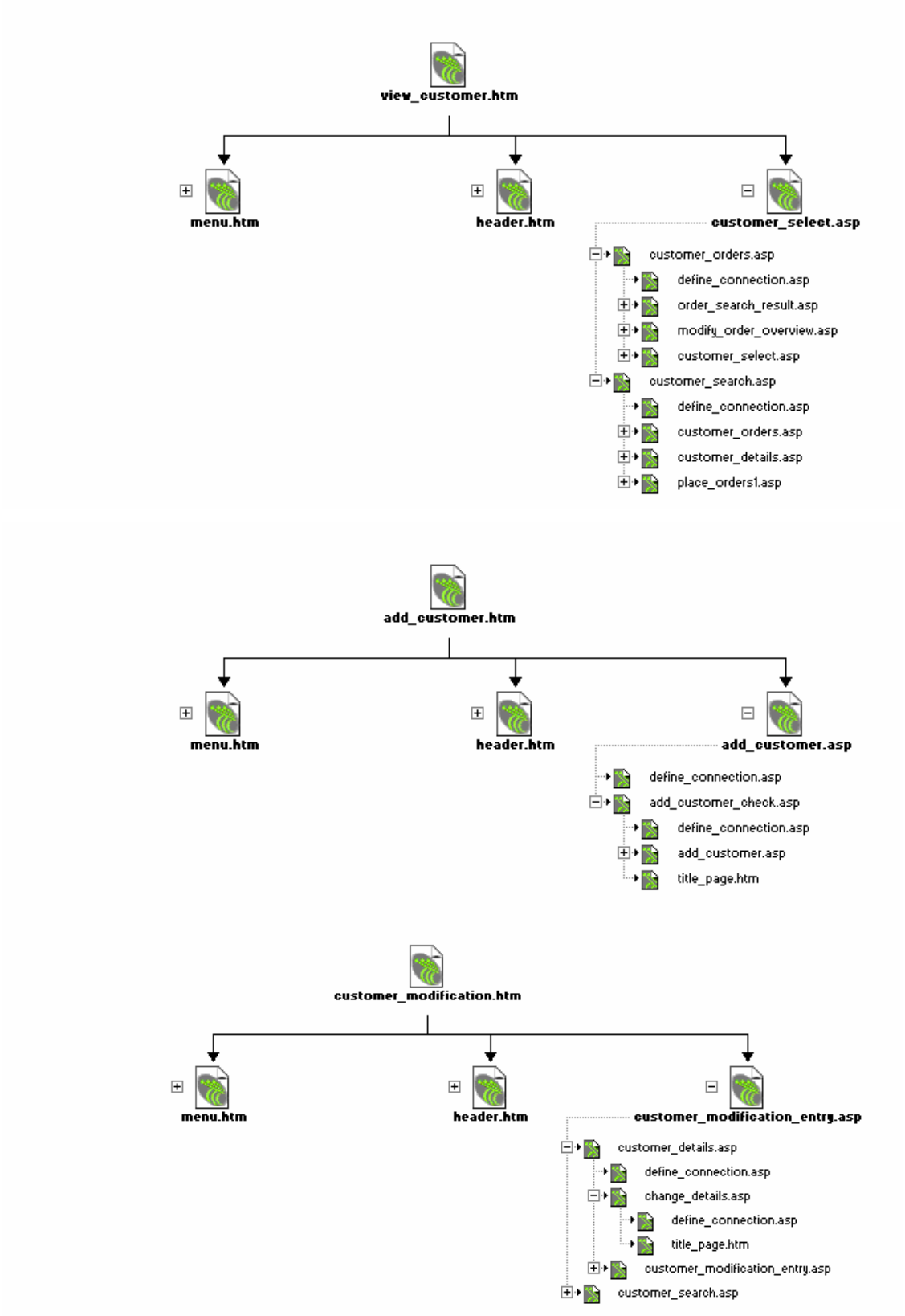

view_article.htm
menu.htm
header.htm
article_search.asp
article_search_result.asp
define_connection.asp
article_search.asp
show_article.asp
define_connection.asp
article_search_result.asp


view_order.htm
menu.htm
header.htm
order_search.asp
order_search_result.asp
define_connection.asp
customer_orders.asp
order_search.asp

Appendix C


place_orders.htm
menu.htm
header.htm
place_orders1.asp
define_connection.asp
place_orders2.asp
define_connection.asp
add_article.asp
define_connection.asp
place_orders2.asp
confirm_order.asp
define_connection.asp
finish_order.asp
define_connection.asp
title_page.htm
add_article.asp
cancel_order.asp
define_connection.asp
confirm_cancel.asp
define_connection.asp
title_page.htm
modify_order_overview.asp
add_article.asp
customer_search.asp
order_modfication.htm
menu.htm
header.htm
order_modification_entry.asp
modify_order_overview.asp
define_connection.asp
remove_article.asp
update_article_qty.asp
define_connection.asp
update_article_qty2.asp
modify_add_article.asp
define_connection.asp
modify_add_article2.asp
modify_order_overview.asp
cancel_order.asp
customer_orders.asp
order_modification_entry.asp

Appendix C


view_customer.htm
menu.htm
header.htm
customer_select.asp
customer_orders.asp
define_connection.asp
order_search_result.asp
modify_order_overview.asp
customer_select.asp
customer_search.asp
define_connection.asp
customer_orders.asp
customer_details.asp
place_orders1.asp
add_customer.htm
menu.htm
header.htm
add_customer.asp
define_connection.asp
add_customer_check.asp
define_connection.asp
add_customer.asp
title_page.htm
customer_modification.htm
menu.htm
header.htm
customer_modification_entry.asp
customer_details.asp
define_connection.asp
change_details.asp
define_connection.asp
title_page.htm
customer_modification_entry.asp
customer_search.asp

Appendix D – System Testing

User Acceptance Testing

The user acceptance was carried out using two dummy users who had no prior knowledge of the system. Initially I explained to them the purpose of the system so they could get a feel for what it was all about. I then showed the system to them, just the main screens, not any of the specific functionality. The users were provided with a copy of the user manual, see appendix F, and then asked to perform specific tasks. The tasks they were asked to perform relate directly to the user requirements outlined in section 3.3. For example the users were asked to view stock details for a specific article and place an order onto the system containing for a specific customer that contained specific stock. The users were also asked to deliberately enter invalid information, such as trying to create an order for a customer that didn't exist or trying to add invalid stock details to an order they had created. In each case the users were asked to provide feedback about how they felt the system performed, how easy it was to perform a task and how user friendly they felt the functional process was.

System Testing

While the vast majority of the testing was undertaken along side implementation and would have proved far too difficult to document due to the extent of it, I did still formulate a test plan which could be used to check the system after the implementation was completed.

Appendix D

System Test Plan

Page	Event	Expected result	Actual Result	Pass/Fail
main_page.htm	Open page	Page loads correctly		
header.htm	Click on *Main* link	Opens main_page		
Menu.htm	Click on *article* link	Opens view_article.htm		
	Click on *order* link	Opens view_order.htm		
	Click on *customer_orders* link	Opens view_customer.htm		
	Click on *place_orders* link	Opens place_orders.htm		
	Click on *update_orders* link	Opens orders_modification.htm		
	Click on *update_customer_details* link	Opens customer_modification.htm		
	Click on *add_customer* link	Opens add_customer.htm		
view_article.htm	Open page	Page loads correctly with article_search.asp in center		
Article_search.asp	Enter invalid article ID	Error Message generated		
	Enter valid article ID	Opens article_search_results.asp and diplays info for that article		
	Click on *Search for article* link	Opens show_article.asp		

Page	Event	Expected result	Actual Result	Pass/Fail
Show_article.asp	Press *Select* with no article selected	Open article_search.asp with error message		
	Select an article and press *Select*	Opens article_search_results		
Article_search_results.asp	Page opens	Displays information for previously selected article		
	Click on the *Back* link	Return to article_search.asp		
View_order.htm	Open page	Page loads correctly with order_search.asp in center		
Order_search.asp	Enter valid order ID	Error message generated		
	Enter valid order ID	Opens up order_search_result.asp		
Order_search_result.asp	Open page	Displays information for entered order ID		
	Click on *Back* button if entered screen from order_search.asp	Return to order_search.asp		
	Click on *Back* button if entered screen from customer_orders.asp	Return to customers orders		
View_customer.htm	Open Page	Page loads correctly with customer_select.asp in center		

Page	Event	Expected result	Actual Result	Pass/Fail
Customer_select.asp	Enter invalid customer details	Error message generated		
	Enter valid customer details	Opens up customer_orders		
	Click on the *Search for Customer* link	Opens up customer_search.asp		
Customer_search.asp	Open page (regardless of what page entered from)	All customers displayed in alphabetical order		
	Press *Select* with no customer selected	Return to page entered screen from with error message generated		
	Select customer and press *Select* ➢ From customer_select.asp ➢ From place_orders1.asp ➢ From customer_modification.asp	Results: Opens customer_orders.asp Returns to place_orders1.asp Opens customer_details.asp		
Customer_orders.asp	Open page	Displays all orders for selected customer		
	Click on an *article ID* link	Opens order_search_results.asp		
	Click on *Modify* link	Opens modify_order_overview.asp		
	Click on *back* button	Return to customer_select.asp		

Page	Event	Expected result	Actual Result	Pass/Fail
Place_orders.htm	Open page	Page loads correctly with place_orders1.asp in center		
Place_orders1.asp	Enter invalid customer ID	Error message generated		
	Enter invalid date (i.e past date)	Error message generated		
	Click on *Search* link	Opens customer_search.asp		
	Enter valid data and click *Next*	Opens place_orders2.asp		
Place_orders2.asp	Open page	Display order ID assigned to the new order and the company for who the order is for. Record created in orders_t1		
	Enter invalid article ID, size or quantity and click *Add to Order*	Error message generated		
	Enter valid data and click add to order	Open add_article.asp with new stock added to the list (if its in list then insert into orders_t2 must have worked)		
	Click *View Order* button	Open add_article.asp and display order information		

Page	Event	Expected result	Actual Result	Pass/Fail
Add_article.asp	Open page	Display stock information fro the current order being processed		
	Click *Add Another Article* link	Open place_orders2.asp		
	Click *Complete Order* link	Open confirm_order.asp		
	Click *Cancel Order* link	Open cancel_order.asp		
Confirm_order.asp	Open Page	Page loads correctly		
	Click on the *Yes* link	Open finish_order.asp		
	Click on the *No* link	Open add_article.asp		
Finish_order.asp	Open page	Page loads correctly		
	Click on the *main menu* button	Title_page.asp opens		
Cancel_order.asp	Open Page	Page loads correctly		
	Click on the *Yes* link	Open confirm_cancel.asp Data for the order deleted from the database		
	Click on the *No* link From add_article From Modify_order_overview.asp	Open: add_article.asp or Modify_order_overview.asp		
Confirm_cancel.asp	Open page	Page loads correctly		
	Click on the *main menu* button	Title_page.asp opens		

Page	Event	Expected result	Actual Result	Pass/Fail
Order_modification.htm	Open page	Page loads correctly with order_modification_entry.asp in center		
order_modification_entry.asp	Enter invaled order ID	Error message displayed		
	Enter valid order ID	Open modify order_overview.asp		
Modify_order_overview.asp	Open page	Page loads correctly		
	Click *Return* button	Open order_modification_entry.asp		
	Click *Cancel Order* Button	Opens cancel_order.asp		
	Click on *Add Article* Button	Opens modify_add_article.asp		
	Click on *Remove Article* Button	Article removed from the database remove_article.asp and the directed back to same page		
	Click on *Update Qty* Button	Opens update_article_qty.asp		
Modify_add_article.asp	Page open	Page loads correctly		
	Click on *cancel* button	Return to modify_order_overview.asp		
	Enter invalid article/qty information	Error msg generated		

Page	Event	Expected result	Actual Result	Pass/Fail
	Enter valid data and *Add to Order*	Article assigned to order in database and open modify_order_overview.asp		
update_article_qty.asp	Open page	Page loads correctly		
	Enter invalid qty and *Submit*	Error message displayed		
	Enter valid qty and *Submit*	Update made to database Open modify_order_overview.asp		
Customer_modification.htm	Open page	Page loads correctly with customer_modification_entry.asp in center		
customer_modification_entry.asp	Enter invalid customer details	Error message generated		
	Enter valid customer details	Open customer_details.asp		
	Click *Search for Customer* link	Open customer_search.asp		
Customer_details.asp	Open Page	Page loads correctly		
	Click *Back Button*	Open customer_modification_entry.asp		
	Submit without filling in all required fields	Error message generated		
	Submit with valid information	Update values in database and open change_details.asp		

Page	Event	Expected result	Actual Result	Pass/Fail
Change_details.asp	Open Page	Page opens correctly		
	Click on *Main Menu* link	Open title_page.asp		
Add_customer.htm	Open Page	Page opens corretly		
	Submit details where all required fields not filled in	Error message generated		
	Submit details where all necessary fields have a value	Data inserted into database and open add_customer_check.asp		
Add_customer_check.asp	Open page	Page opens correctly displaying the newly entered information		
	Click on *main menu* link	Open title_page.htm		

Appendix E – Evaluation

The developed prototype was evaluated against the following criteria using the results from the system testing, the user acceptance testing and my own reflection as the developer.

1. Correctness

The prototype includes all the minimal requirements defined in section 3.3 as well the additional feature of being able to view orders by customer. Taking into consideration that the users don't fully understand the system I feel this point is best proved by the system test plan as the full run through allows the user to perform the various required tasks.

2. Reliability

Due to the nature of the continual testing throughout the development I was confident that the system would prove to be reliable. All the functions of the system worked without failure throughout the execution of the system test plan and the system user acceptance testing. At all times data was successfully recorded, deleted or updated within the database as it should have been, and the navigation and display of the web pages was consistent and correct throughout.

3. Efficiency

Although no formal procedures were followed here to record to see if the developed system improves upon the current paper based filing system, I am extremely confident that it does. If the dummy users general attitude to the system was good and they believed that the ease of use for performing specific actions to be of reasonable quality then surely the actual users would be able to appreciate the benefits that the system provides. The time taken to find order details on the new system is in seconds, something which definitely can't be said about the current paper based filing system.

4. Usability

The general feedback from the users was that the system was very usable in that it had a 'clear structure and layout', with the menu option making for 'easy movement between pages'. One bad point that was picked up on was the bad use of colour and wording. On several pages, such as the customer_select.asp page the title was in the same colour as a link from that page and not only that but the wording was similar which was slightly misleading. The users said they could tell the

difference as the link is underlined but they had to look twice and that makes for poor readability due to the increase in cognitive overhead.

Another point that the users didn't pick up on but I did was to do with the mis-alignment of the table on the customer_orders.asp page, something I know to be caused at the programming level by the use of hidden fields within the page, but not something I was able to eradicate in time.

A final issue was noticed in the demonstration of this software and is to do with when users place orders, they are expected to know the articles ID's and their sizes. This is perhaps an oversight as I assume too much and possibly a way forward would be to link the orders page to the article search page so the users are able to select article codes from here.

Some very positive feedback was given by the users about the use of error handling. The users found, as I did in testing that entering invalid data always generated an appropriate error, so the robustness of the system is good, but more than that the users acknowledged the wording, colour and positioning of the error message displayed as it 'brought the error straight to the users attention' and was worded do that the 'problem was easily recified'.

5. Maintainability

Due to the careful planning that went into the systems design it should be easy to see where any problems originate from. The chosen development tools were ones which Blacks have readily available so any software updates can be made with minimal disruptions.

Appendix F – User Manual

Contents

1. Start up……..**2**
1.1 Main Page……...**2**
1.2 System Functionality…….**2**
2. Viewing Stock Article Details……..**3**
2.1 Article ID Entry…….**3**
2.2 Article ID Search……...**3**
3. Viewing Individual Order Details……...**5**
3.1 Order ID Entry……..**5**
4. Viewing Order Details by Customer.....**6**
4.1 Customer Information Entry……...**6**
4.2 Customer Search……...**7**
5. Placing Orders……..**8**
5.1 Creating an Order…….**8**
5.2 Adding Articles to the order…….**8**
5.3 Completing the Order……...**9**
5.4 Cancelling the Order……**10**
6. Modifying an Order……**11**
6.1 Order Selection……..**11**
6.2 Updating Article Quantities……**11**
6.3 Adding an Article……..**12**
6.4 Removing an Article…….**12**
6.5 Cancelling the Order……**13**
7. Modifying Customer Details……...**14**
7.1 Customer Selection……**14**
7.2 Updating the Details……..**14**
8. Adding New Customers Details……...**15**

Appendix F

1 – Start Up

1.1 Main Page

Upon start up you will be presented with the following screen:

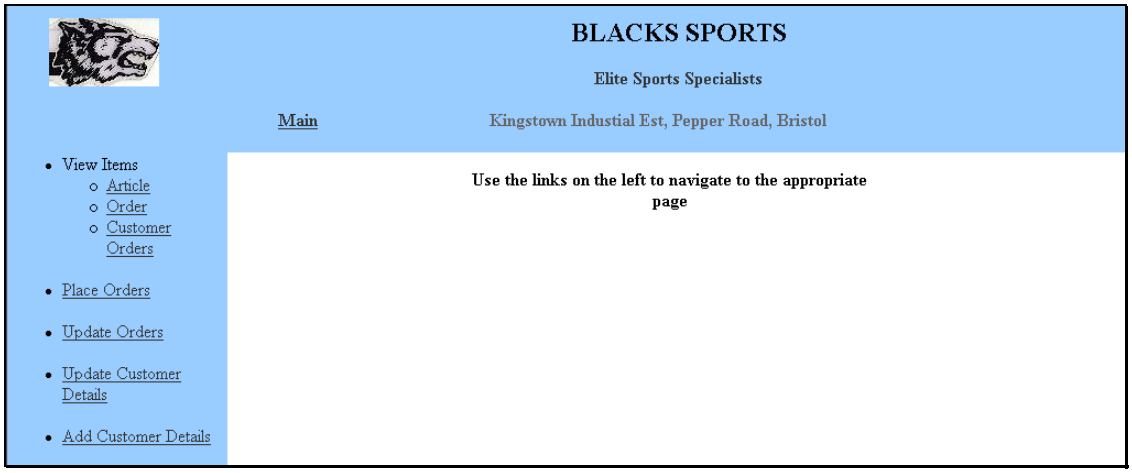


Figure 1

The layout of the screen is consistent throughout the system. It consists of the options menu down the left hand side of the page, the company header displaying company information and the main section of screen where all of the appropriate pages required to perform the requested action will appear when you click on one of the links from the menu.

1.2 System Functionality

By using the menu links on the left you can navigate to different parts of the site and perform the following functions:

- View Article Stock Details
- View Individual Order Details
- View Order Details by Customer
- Place a New Order
- Modify an Existing Order
- Modify Existing Customer Details
- Add a New Customer to the System

Appendix F

2 – Viewing Article Stock Details

From the main menu, click on the *article* link under the *view items* icon. Within the central part of the screen you will be presented with the following, *Search by Article Number* page:

Figure 2

2.1 Article ID Entry

To view the stock information for an article enter the article ID into the data field provided and click on the submit button, which will open the A*rticle Search Results* page displaying the article information, as shown in figure 3. If you enter an invalid article ID then you will be presented with an error message, and you will have to enter the article ID again.

Figure 3

The *back* button at displayed at the bottom of the order information on the *Article Search Results* page can be clicked on to return to the *Search by article number* page.

2.2 Article ID Search

If the article ID is unknown then you can use the article search function by clicking on the *Search for article* link as displayed in figure 2 then a screen displaying all articles will be displayed, as shown in figure 4. All you need to do is select the required article by highlighting the appropriate radio button and click on the *select* button which will open up the *Article Search Results* page.

Appendix F

Figure 4

Appendix F

3 – Viewing Individual Order Details

From the main menu, click on the *order* link under the *view items* icon. Within the central part of the screen you will be presented with the following, *Search by order number* page:

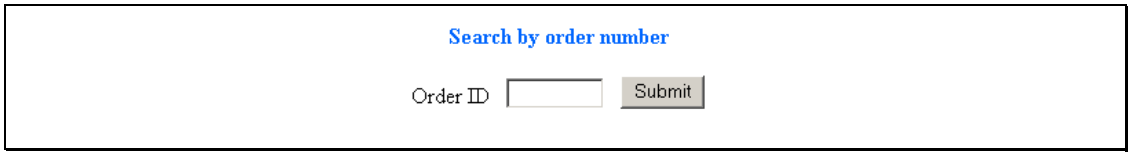


Figure 5

3.1 Order ID Entry

Enter an order ID into the data field provided and click on the *submit* button. If a valid order ID is submitted then the *Order ID Search Results* page will open up displaying the information for the submitted order ID, as shown in figure 6. If the submitted order ID is invalid then an error message will be displayed and an order ID must be entered again.

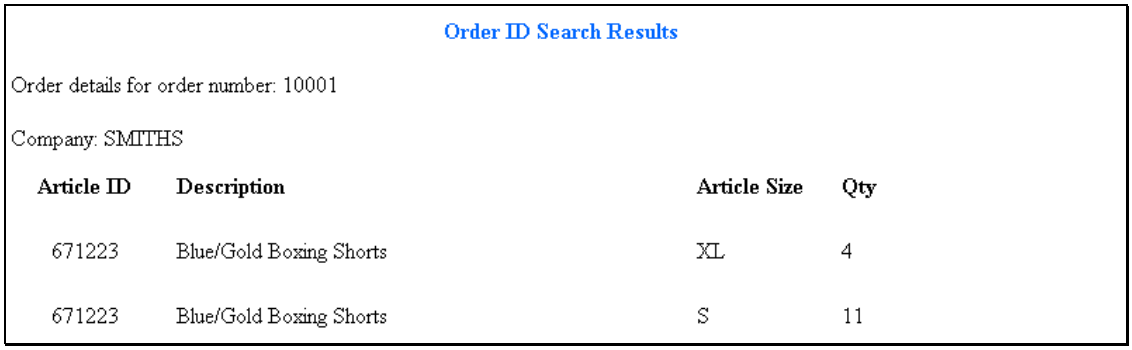


Figure 6

The *back* button at displayed at the bottom of the order information on the *Order ID Search Results* page can be clicked on to return to the *Search by order number* page.

Appendix F

4 – Viewing Order Details by Customer

From the main menu, click on the *Customer Orders* link under the *view items* icon. Within the central part of the screen you will be presented with the following, *Search By Customer* page:

Figure 7

4.1 Customer Information Entry

This screen allows you to enter data in one of two ways, by customer ID or by customer name. Simply select which entry method you want to use from the drop down menu, enter the customer ID or name into the data field provided and click on the *submit* button. If a valid customer identity is submitted then the *Customer Orders* page will open up displaying the orders for that customer, as shown in figure 8. If the submitted customer identity is invalid then an error message will be displayed and the information must be entered again.

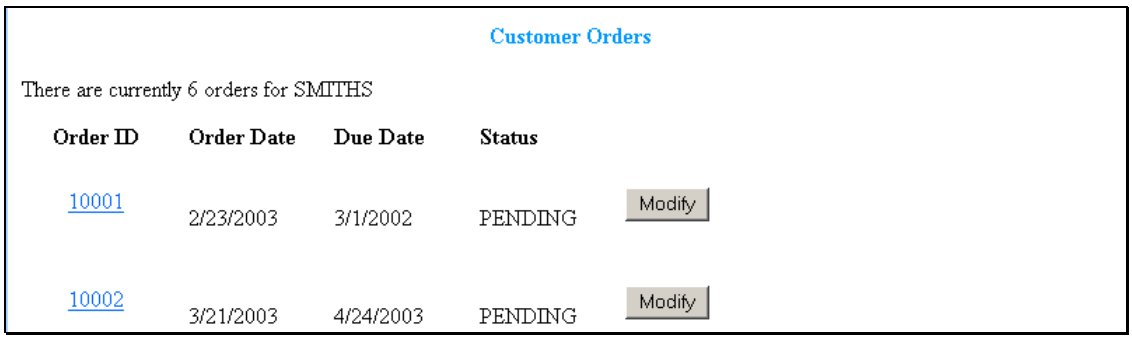


Figure 8

From the *Customer Orders* page you can click on an order ID on the left to open up the *Order ID Search Results* page, which will display the article information for that order. You can also use the *Modify* button on the right of an order, as displayed in figure 8, to go to the order modifications page and change the order details. Order Modification is covered in section 6 of this manual. The *back* button at displayed at the bottom of the list of customer orders can be clicked on to return to the *Search by Customer* page.

Appendix F

4.2 Customer Search

If you wish to search for a customer then you can use the customer search function by clicking on the *Search for Customer* link as displayed in figure 7. A screen displaying all customers in alphabetical order will be displayed, as shown in figure 9. All you need to do is select the desired customer by highlighting the appropriate radio button and click on the *select* button which will open up the *Customer Orders* page, displaying all orders for the selected customer.

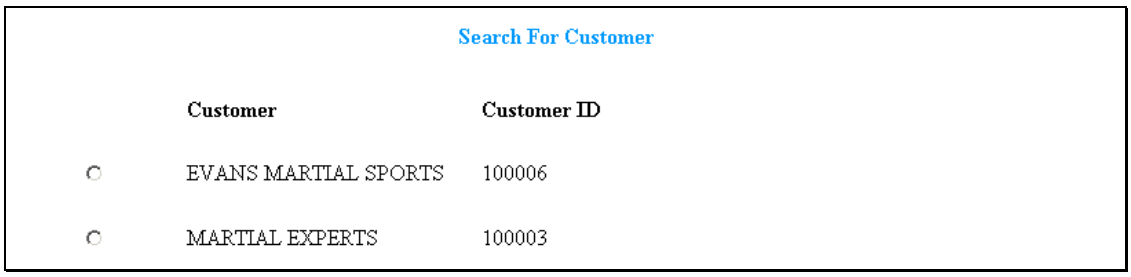


Figure 9

Nb:

This search page can also be accessed from the *Place Orders* page and the *Customer Modification* page.

If the page is accessed from the *Place Orders* page then when you select a customer you will be directed back to *Place Orders* page.

If the page is accessed from the *Customer Modification* page then when a customer is selected you will be directed to the *Customer Details* page which will then display all information for that selected customer.

Appendix F

5 – Placing Orders

5.1 Creating an order

From the main menu, click on the *Place Orders* link under the *view items* icon. Within the central part of the screen you will be presented with the following, *Place Orders* page:

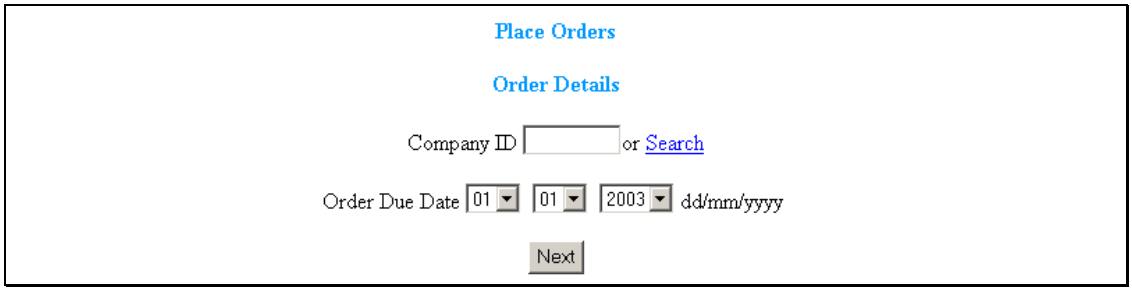


Figure 10

To create an order you must specify which customer it is for and provide a delivery due date for the order. To do this you should provide the details in the data fields provided, as shown in Figure 10, and click on the *Next* button to advance to the next page. If you provide an invalid company ID or date, such as a past date, then an error message will be displayed on the screen. If you don't know the customer ID then you can click on the *Search* link which will take you to the *Search for Customer* page where you can select the desired customer and return to this page, at which point the Company ID field will automatically be populated for you.

5.2 Adding Stock to an Order

On the next page, the *Add Articles* page (Figure 11), you will see that a unique order ID has been generated for the order and is displayed along with the name of the Customer for whom the order is for. It is from this page that users can add stock to the order.

To do this you must enter an article ID, the article size and the quantity of that article required, in the data fields provided and then click on the *Add to order* button. If you enter an invalid article ID, an invalid size for that article or an invalid quantity then an error message will be displayed.

If you enter valid information then the *Article Detail Check* page shown in figure 12 will open and from here there are a number of options. By pressing the appropriate button you can complete the order, cancel the order or add more articles to the order, in which case you will be directed back to the *add articles* page.

Also from the *Add Articles* page there is a *View Order* button which opens up the *Article Detail Check* page so you can see what is currently on the order.

Appendix F

Figure 11

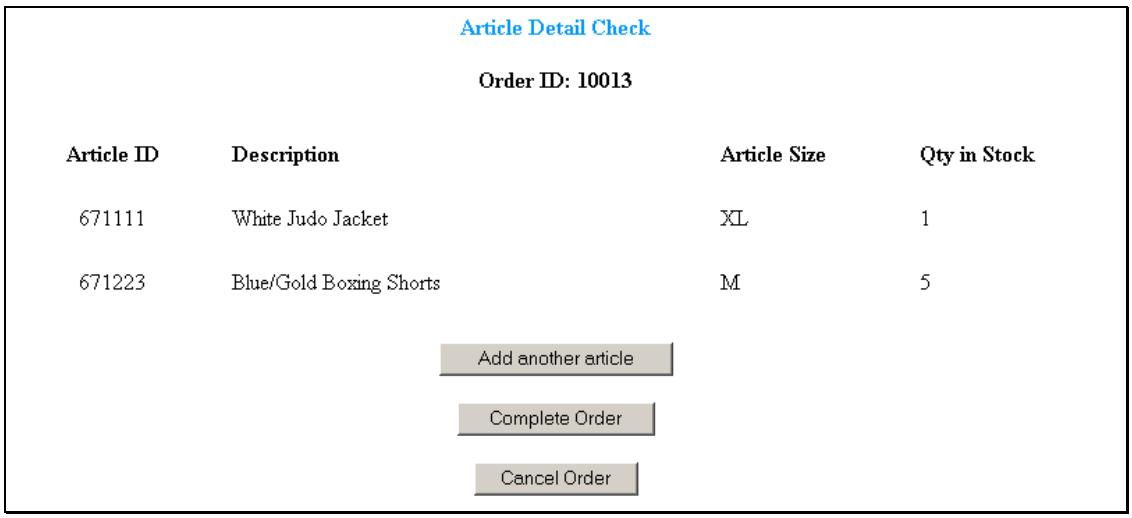

Article Detail Check
Order ID: 10013
Article ID Description Article Size Qty in Stock
671111 White Judo Jacket XL 1
671223 Blue/Gold Boxing Shorts M 5
Add another article
Complete Order
Cancel Order

Figure 12

5.3 Completing an Order

Once all articles have been added to an order and you wish to complete you need to click on the *Complete Order* button from the *Article Detail Check* page as shown in Figure 12. You will then be asked to confirm the completion of your order at which point you can either click *Yes* or *No*, see figure 13. Clicking *Yes* confirms the order and clicking *No* will direct you back to the *Article Detail Check* page. Once an order has been confirmed a page will display that contains a message

telling you the order was successfully recorded within the system and a button to press to return to the main menu, see figure 14.

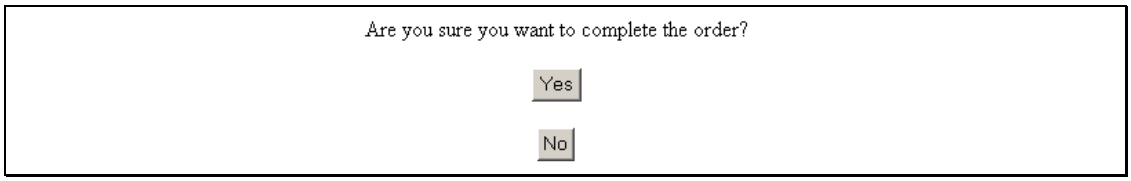


Figure 13

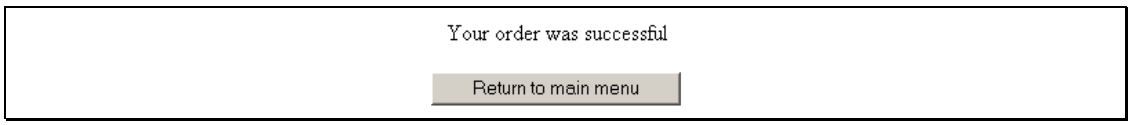


Figure 14

5.4 Cancelling the Order

If you decide that you don't want to continue processing the order then you need to click on the *Cancel Order* button from the *Article Detail Check* page as shown in Figure 12. You will be presented with a page asking you to confirm the cancellation of the order, see figure 15. Clicking on *No* will return you to the *Article Detail Check* page and clicking *Yes* will confirm the cancellation of the order, resulting in a page being displayed that contains a message telling you the order was successfully cancelled and a button to press to return to the main menu, see figure 16.

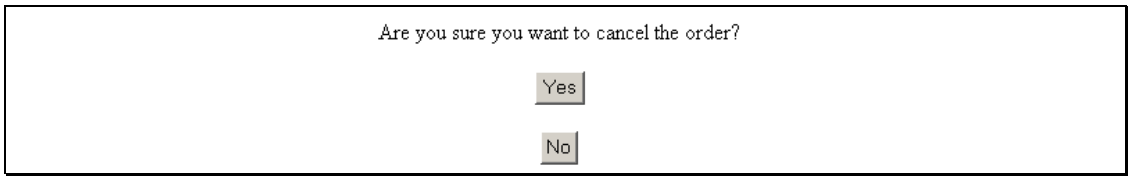


Figure 15

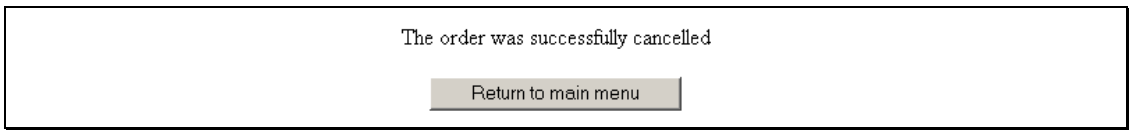


Figure 16

Appendix F

6 – Modifying an Order

It may be necessary to modify existing order details for whatever reason and this section explains the processes involved.

6.1 Order Selection

From the main menu, click on the *Update Orders* link under the *view items* icon. Within the central part of the screen the *Order Modification* page will be displayed. It is identical in layout to that of the *Search by order number* page, shown in figure 5. All you need to do is enter the ID of the order you wish to modify and click on the submit button and you will be directed to the *Order Overview* page, figure 17. An error message will be displayed if the entered ID is invalid.

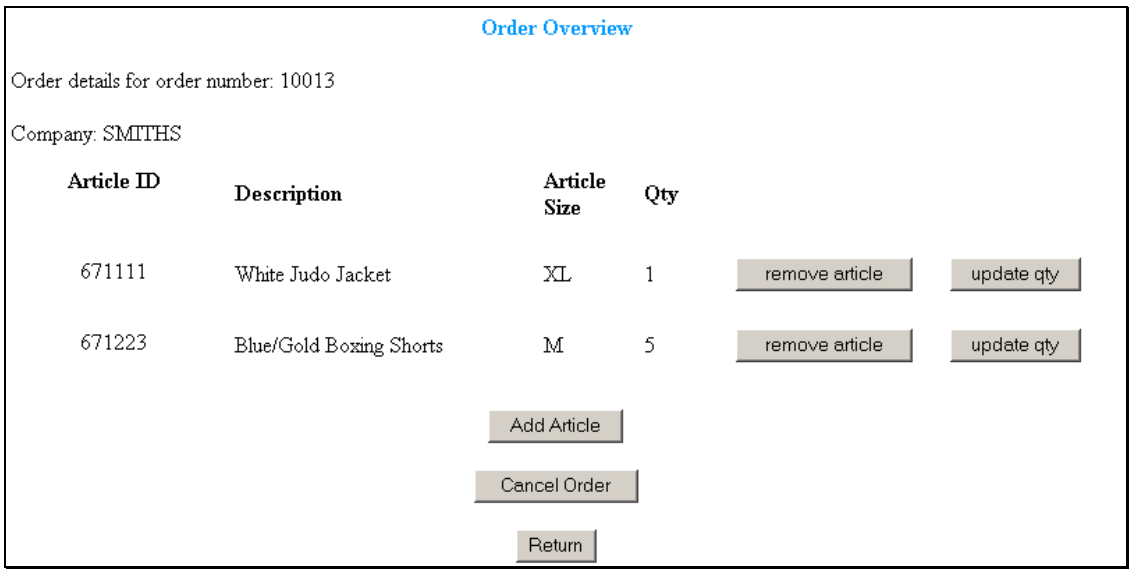


Figure 17

The page lists all the stock currently assigned to an order and a number of options for modifying the order. The *return* button at the bottom of the page can be used to exit from this page when you are satisfied with the modifications you have made.

6.2 Updating Article Quantities

To update the quantity of a particular item assigned to the order click on the appropriate *update qty* button adjacent to the item. A page such as the one in figure 18 will be displayed, listing the article and the current quantity assigned to the order. To adjust the quantity you need to enter the new value in to the data field and click on the *submit* button. You will be redirected back to the

Order Overview page where the new information will be detailed. Entry of an invalid quantity (e.g 0) will result in an error message.

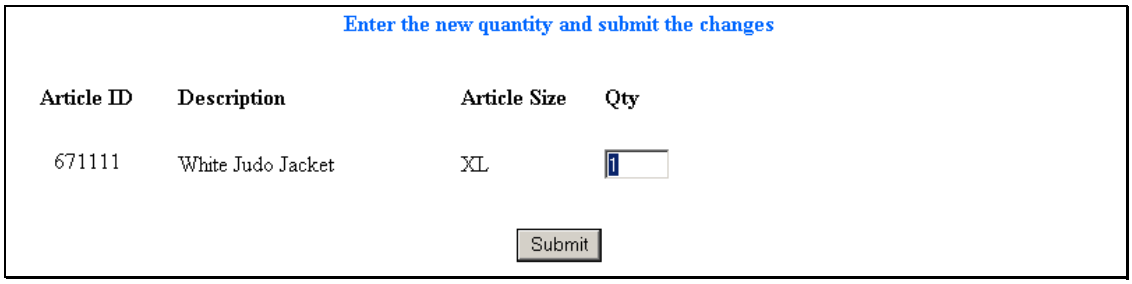


Figure 18

6.3 Adding an Article

To add a new article to the order you need to click on the *Add Article* button, shown if figure 17. You will be presented with the following page:

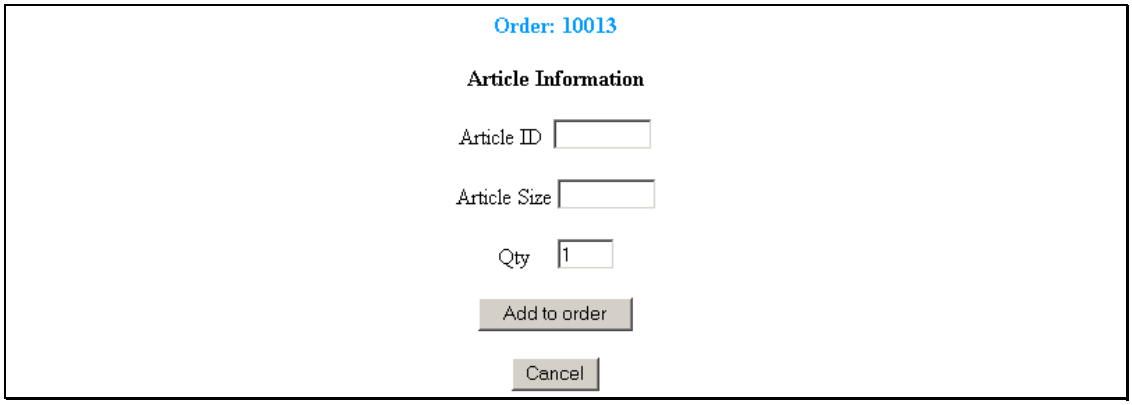


Figure 19

To add an article you need to fill in the Article ID, Article Size and Qty fields and then click on *Add to Order* as shown in figure 19. If any of the data entered is invalid an error message will be displayed. If not then you will return to the *Order Overview* page that will now detail the new article information.

If you don't wish to add another article to the order then click on the *Cancel* button and you will return to the *Order Overview* page without making any changes.

6.4 Removing an Article

To remove an article from the order click on the *remove article* button, as shown in figure 17, which is next to the article you want to remove. The article will be removed from the order automatically and the page will refresh to show this.

6.5 Cancelling the Order

To cancel the order, click on the *Cancel Order* button from the *Order Overview* page, as shown in figure 17. You will then be asked to confirm the cancellation of the order, see figure 15. If you click *No* you will return to the *Order overview* page, but if you select *Yes,* the order will be cancelled and a page will display a message to this effect and prompt you to return to the main menu, see figure 16.

Appendix F

7 – Modifying Customer Details

From the main menu, click on the *Update Customer Details* link under the *view items* icon. Within the central part of the screen the *Customer Modification* page will be displayed, figure20.

Figure 20

7.1 Customer Selection

The process of selecting the customer here is identical to that explained in Section 4 of this handbook. You can either enter a customer ID or a customer name and click submit or you can click on the *Search For Customer* link, in order to proceed to the *Customer Details* page, shown in figure 21. See section 4.2 for information regarding the Customer Search Option. If you enter an invalid customer identity then an error message will be displayed.

7.2 Updating the Details

Once you have selected the customer whose details you wish to modify you will be presented with the following page, which will contain the information regarding the chosen customer.

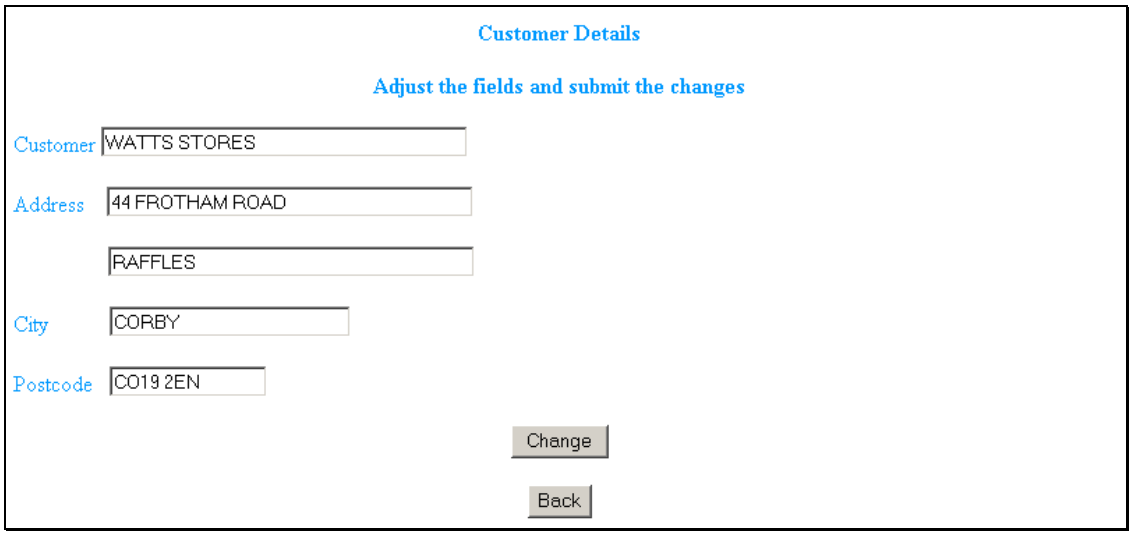


Figure 21

Appendix F

To modify the company details you need to adjust the current data displayed within the fields and click on the *Change* button. If you decide not to change the details, clicking the *Back* button will return you to the *Customer Modification* page. You must ensure that all fields, except from the second address field which is optional, are supplied with a value otherwise an error message will be generated. If the changes are successful then the following page will be displayed and you can return to the main menu:

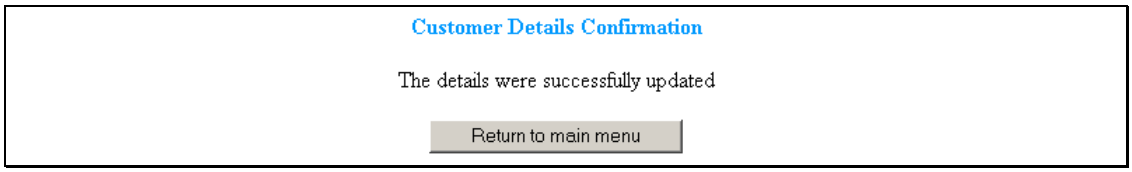


Figure 22

Appendix F

8 – Adding New Customer Details

From the main menu, click on the *Add Customer Details* link under the *view items* icon. Within the central part of the screen the *Customer Modification* page will be displayed, figure23. You must supply values for each field except for the second address field as it is optional. Once you are satisfied that the data is correct you should click on the *Submit* button to enter the details on to the database. If they are valid then a page will appear, which displays the details that were added onto the database and a message to notify you of the addition of the details. A button is also on the page so you can return to the main menu, see figure 24.

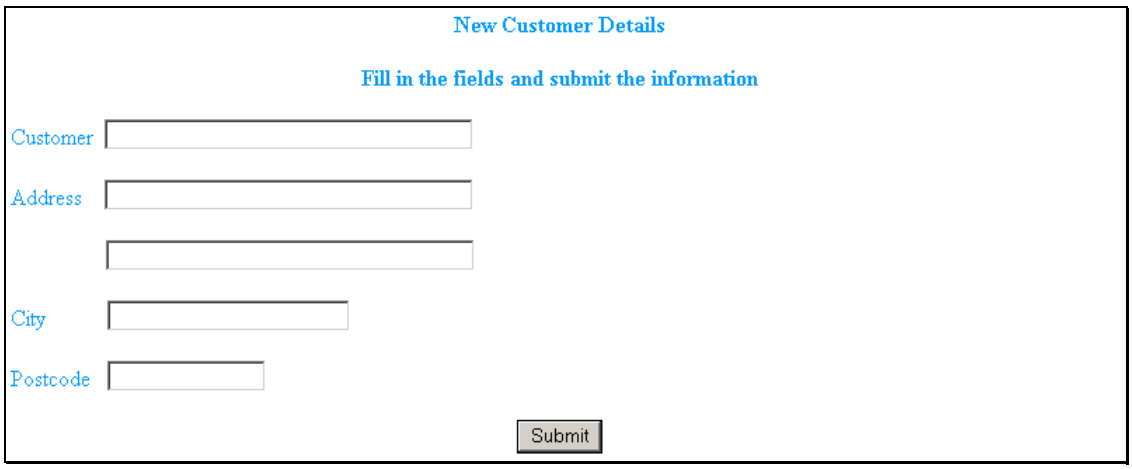


Figure 23

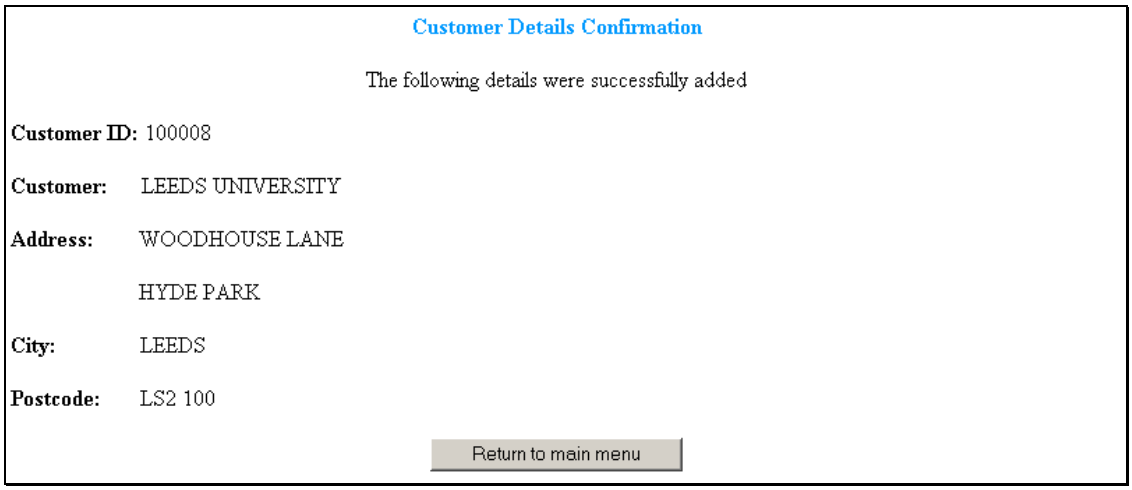


Figure 24